INRIA
CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél. (3) 954 90 20

Rapports de Recherche

N° 430

THE PARTITION MODEL : A FUNCTIONAL APPROACH

Nicolas SPYRATOS

Juillet 1985

THE PARTITION MODEL : A FUNCTIONAL APPROACH

Nicolas SPYRATOS
Université de Paris-Sud
91405 Orsay - FRANCE

Abstract - We study the partition model introduced in [7] from a functional viewpoint, whereby attributes are modeled by special functions mapping the set of nonnegative integers into its power set. We then use the partition model in order to study database equivalence and, more specifically, database decomposition.

Résumé - Nous étudions le modèle de partitions présenté dans [7] d'un point de vue fonctionnel, où les attributs sont modélisés par des applications de l'ensemble des entiers positifs dans son ensemble de parties. Ensuite, nous utilisons le modèle de partitions afin d'étudier l'équivalence des bases de données et, plus particulièrement, la décomposition d'une base de données.

CONTENTS

1. AN INFORMAL INTRODUCTION OF THE MODEL

Most of the formal database studies that are under way at present are concerned with the relational data model introduced by Codd [2]. One notable feature of this model is its being almost devoid of semantics. One approach to remedy this deficiency is to devise means to specify the missing semantics. Mathematical logic provides a formal framework for such an approach. It has been used in query languages, integrity modeling and maintenance, query evaluation, database design through dependencies, representation and manipulation of deduced facts, and incomplete information.

In this paper, we study the partition model introduced in [7] from a functional viewpoint. We assume that the set of all objects that can possibly be of interest is a countable set, say ω. The main innovation in this report is to model attributes of the objects in ω by some special functions mapping ω into the set $P\omega = \{\tau | \tau \subseteq \omega\}$, the set of all subsets of ω. These functions, that we call partitionings, can be "multiplied" and "added". The set of all partitionings turns out to be a lattice, under product and sum of partitionings. The database model introduced in this paper is based on the lattice of partitionings. Some of its underlying concepts were first introduced in [7].

In this section we present an informal introduction of the model using an example. Think of an enterprise, say a small tourist agency. Suppose that, at a given time, the objects of interest to the agency are the four automobiles owned by the agency and the five persons employed by the agency. Moreover, assume the following set of nine integer identifiers : $\varepsilon = \{1,2,3,4,5,6,7,8,9\}$, where 1,2,3, and 4, are the identifiers of the four automobiles of the agency, and 5,6,7,8, and 9, are the identifiers of the five employees of the agency. From now on we confuse objects with their identifiers and we assume that 1,2,...,9 are the objects of interest to the agency. Of course the set ε changes during the life of the enterprise. For example, the agency may acquire additional automobiles, or increase the number of its employees, or even expand to new activities. Here we are interested in a specific moment in the life of the enterprise (in

a "snapshot" of the enterprise), where the nine objects of the set ε, listed above, are the only objects of interest to the enterprise.

Some attributes of the objects in the set ε may be of interest during the operation of the enterprise. Examples of such attributes are the TYPE of an automobile or the DEPOT to which an automobile is assigned, and the RANK or the SALARY of an employee. We assume that TYPE and DEPOT are undefined for employees, and that RANK and SALARY are undefined for automobiles. We may view each attribute of objects in ε as a function associating objects of the set ε with convenient or familiar identifiers, as shown in Figure 1. Of course, two or more objects can be associated with the same identifier, under a given attribute. For example, the objects 1 and 2 are associated both with the identifier "bus", under TYPE, as shown in Figure 1. However, the essential information here is that the objects 1 and 2 are associated with the same identifier. The identifier itself is of lesser interest, as it may depend, for instance, on personal taste (a Greek would rather prefer "λεωφορεῖο" as identifier, instead of "bus"). Thus a more fundamental point of view is the following : each attribute of objects in ε denotes a function, associating each object x in ε with the set of all objects in ε that are associated with the same identifier as x. This point of view is depicted in Figure 2. Thus, the identifier "TYPE" denotes the function T, the identifier "bus" denotes the set {1,2}, the identifier "limousine" denotes the set {3,4}, and the identifier "undefined" denotes the empty set. In this way, we have set up a one-one correspondence between the identifiers in the range of attribute TYPE and the sets in the range of function T, as follows :

bus $\longleftrightarrow$ {1,2}
limousine $\longleftrightarrow$ {3,4}
undefined $\longleftrightarrow$ $\emptyset$

Given an identifier t, we call interpretation of t the set of objects corresponding to t. We denote the interpretation of t by $\imath(t)$.

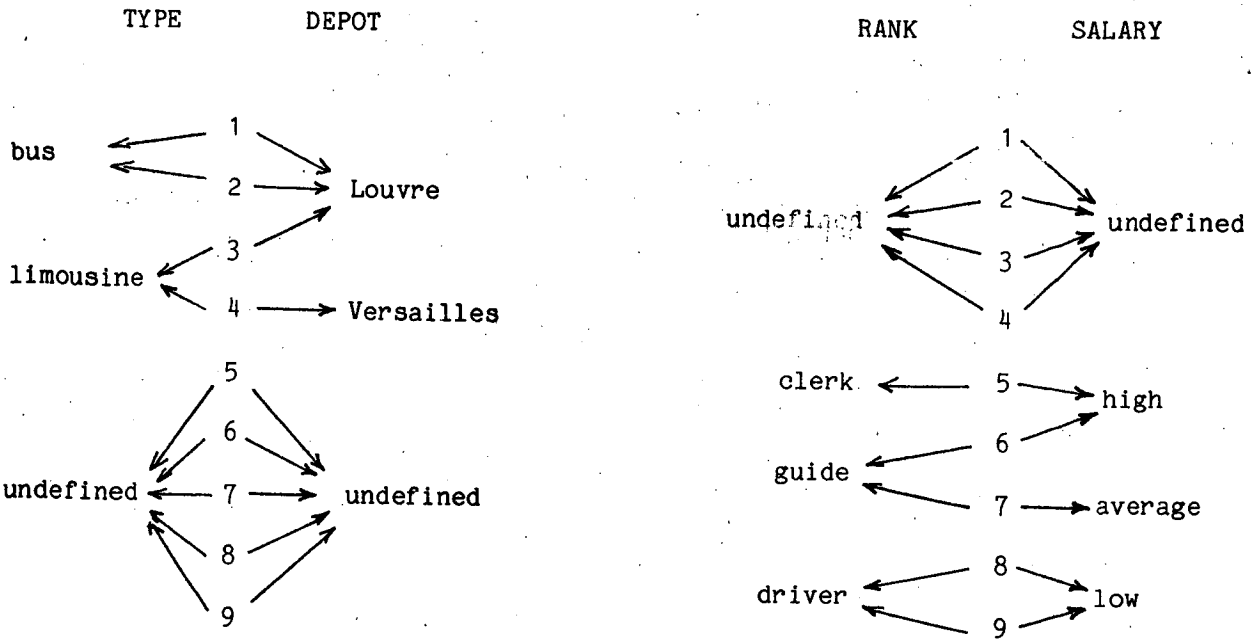


FIGURE 1. Attributes of objects seen as functions associating objects with identifiers

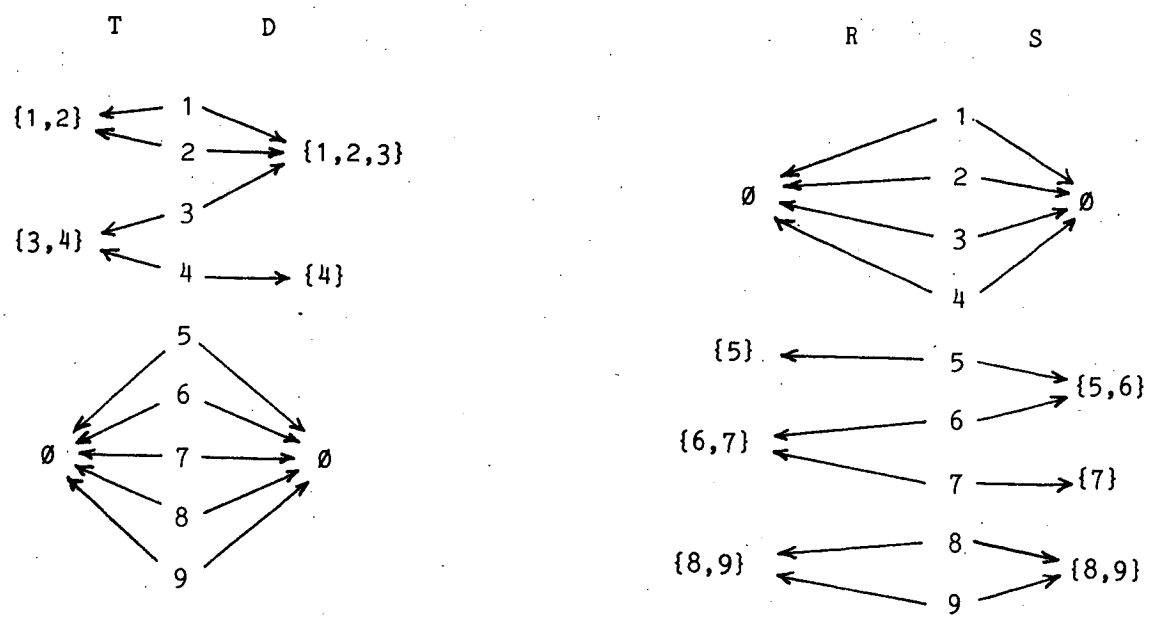


FIGURE 2. Attributes of objects seen as functions associating objects with sets of objects.

Thus ι(bus) = {1,2}, ι(limousine) = {3,4}, and ι(undefined) = $\emptyset$. In the opposite direction, given a set of objects τ, we call name of τ the identifier corresponding to τ. We denote the name of τ by $\nu(\tau)$. Thus $\nu(\{1,2\})$ = bus, $\nu(\{3,4\})$ = limousine, and $\nu(\emptyset)$ = undefined. Actually we can view ν as a function naming sets, and ι as a function interpreting identifiers, as shown in Figure 3.

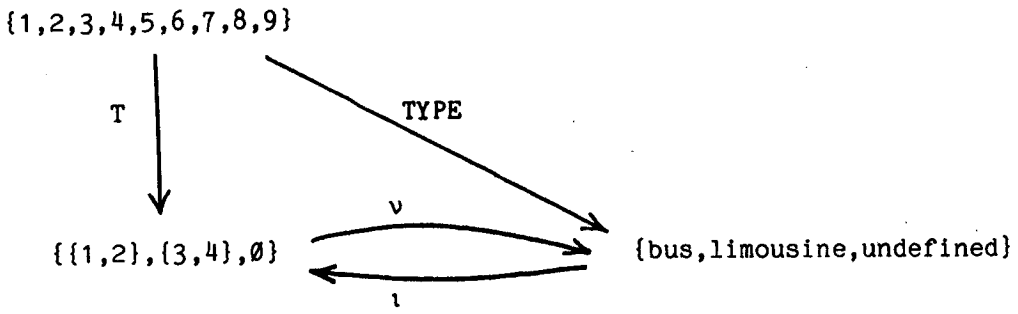


FIGURE 3. Naming of sets and interpretation of identifiers, seen as functions

From this point of view, the attribute TYPE can be obtained by function composition from T and ν, namely $TYPE=\nu_{o}T$. Similarly, the function T can be obtained by function composition from TYPE and ι, namely $T=\iota_{o}TYPE$. We consider the function T as essential information about the objects in ε, whereas the function TYPE is useful but name-dependent information. Let us note that the function T satisfies the following interesting properties (see Figure 2)

(a) $\forall x \in \varepsilon \quad T(x) \neq \emptyset \Rightarrow x \in T(x)$

(b) $\forall x \in \varepsilon \quad \forall y \in \varepsilon \quad T(x) \neq T(y) \Rightarrow T(x) \cap T(y) = \emptyset$

We call partitioning function, or simply partitioning, any function on ε satisfying these properties. In fact all functions of Figure 2 are partitionings. The important property of a partitioning is that given a naming, it is transformed to an attribute (see Figure 3).

Let us return to our example of a tourist agency. Suppose that we would like to define a new attribute, called ASSIGNMENT, that associates each object in ε with its TYPE and its DEPOT. Consider, for example, the object 1 (Figure 1). This object has the same ASSIGNMENT as 2, because 1 and 2 have the same TYPE and the same DEPOT. On the other hand, 2 and 3 have the same DEPOT but different TYPE, so 2 and 3 do not have the same ASSIGNMENT. These examples motivate the following definitions (refer to Figure 2). Two objects x and y are equivalent with respect to T and D, denoted by $x \hat{=} y$, if $T(x) = T(y)$ and $D(x) = D(y)$. The product of T and D, denoted by T.D, is defined as follows :

$$\forall x \in \varepsilon \quad (T.D)(x) = \emptyset, \text{ if } T(x) = \emptyset \text{ or } D(x) = \emptyset$$
$$= \{y \in \varepsilon \mid x \hat{=} y\}, \text{ otherwise} \qquad (1)$$

Let us observe that $T(x) \cap D(x)$ is the set of all objects that have the same TYPE and the same DEPOT as the object x. Thus we can rewrite the definition of the product as follows :

$$\forall x \in \varepsilon \quad (T.D)(x) = T(x) \cap D(x) \qquad (2)$$

For example, referring to Figure 2, we find

$$(T.D)(1) = \{1,2\} \text{ , following (1)}$$
$$= \{1,2\} \cap \{1,2,3\} \text{ , following (2)}$$

These two equivalent representations of (T.D)(x) give rise to two equivalent representations of the product of T and D, as shown in Figure 4. The first representation, denoted by T.D, shows the product before computing intersections. The second representation, denoted by A, shows the product after computing intersections. Of course T.D = A, but there is an important difference between the two representations when it comes to naming them. Indeed, we can name T.D using the identifiers that appear in Figure 1, and a naming convention. For example, let us adopt the following naming convention for nonempty sets (the empty set will always be named by "undefined") :

if μ is a naming of T and ν is a naming of D then the name of the set $T(x) \cap D(x)$ is the identifier $\mu(T(x))|\nu(D(x))$, where the vertical line segment denotes juxtaposition

Thus the name of the intersection $\{1,2\} \frown \{1,2,3\}$ is "bus|Louvre", because the name of $\{1,2\}$ is "bus" and the name of $\{1,2,3\}$ is "Louvre". Proceeding in this manner we can name all the interesections appearing in the range of T.D. The resulting names are shown in Figure 5, and they constitute the range of the function TYPE.DEPOT. Thus we have been able to name the function T.D using old names and a naming convention. However, this is not possible for the function A as the nonempty sets appearing in its range have not been previously named. Thus, if we want to name the function A we must introduce new names for the nonempty sets $\{1,2\}$, $\{3\}$, and $\{4\}$ that appear in its range. Assume the following naming

$\{1,2\} \longleftrightarrow$ groups
$\{3\} \longleftrightarrow$ intown
$\{4\} \longleftrightarrow$ outoftown

Using this naming and the function A we obtain the function ASSIGNMENT shown in Figure 5.

Some important remarks are in order at this point. First, we remark that the product T.D is also a partitioning, as it satisfies the properties (a) and (b) mentioned earlier. Second, although the functions T.D and A are equal, their named representations TYPE.DEPOT and ASSIGNMENT are not. That is, T.D or A carry the essential information, whereas TYPE.DEPOT and ASSIGNMENT are two different name-dependent representations of that information. Third, the function A carries "finer" information than the function T in the sense that, if two objects of ε are associated with the same set under A, then they are associated with the same set under T. This is denoted by $A \leq T$, and it is defined formally as follows

$$A \leq T \text{ iff } \forall x \in \varepsilon \quad A(x) \subseteq T(x)$$

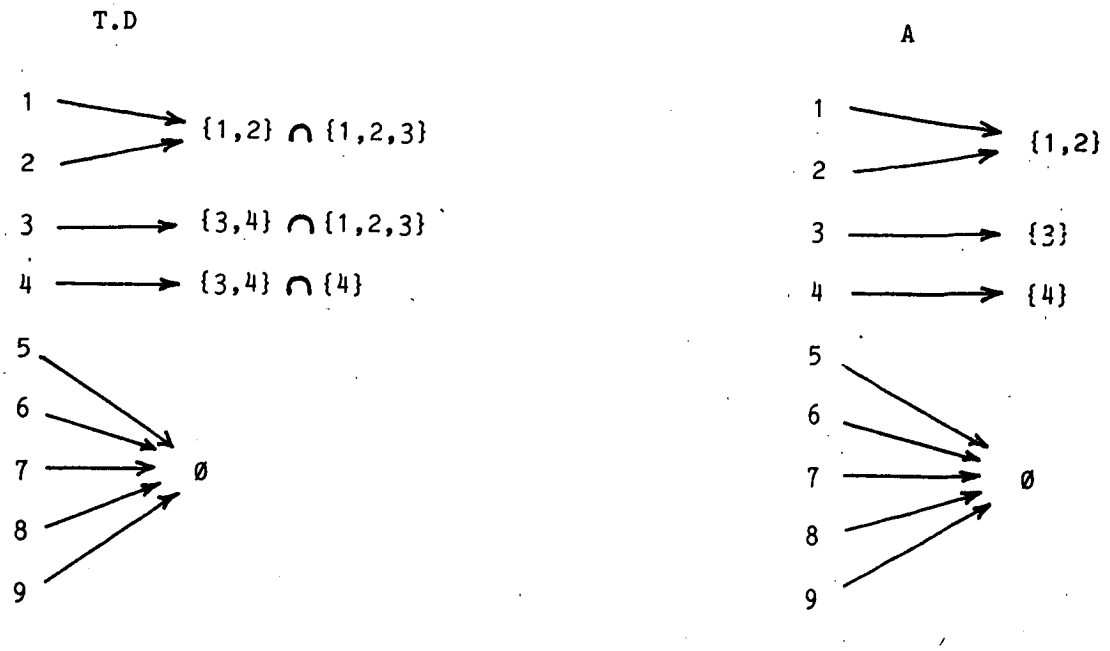


FIGURE 4. The product of T and D before and after computing intersections (denoted T.D and A, respectively).

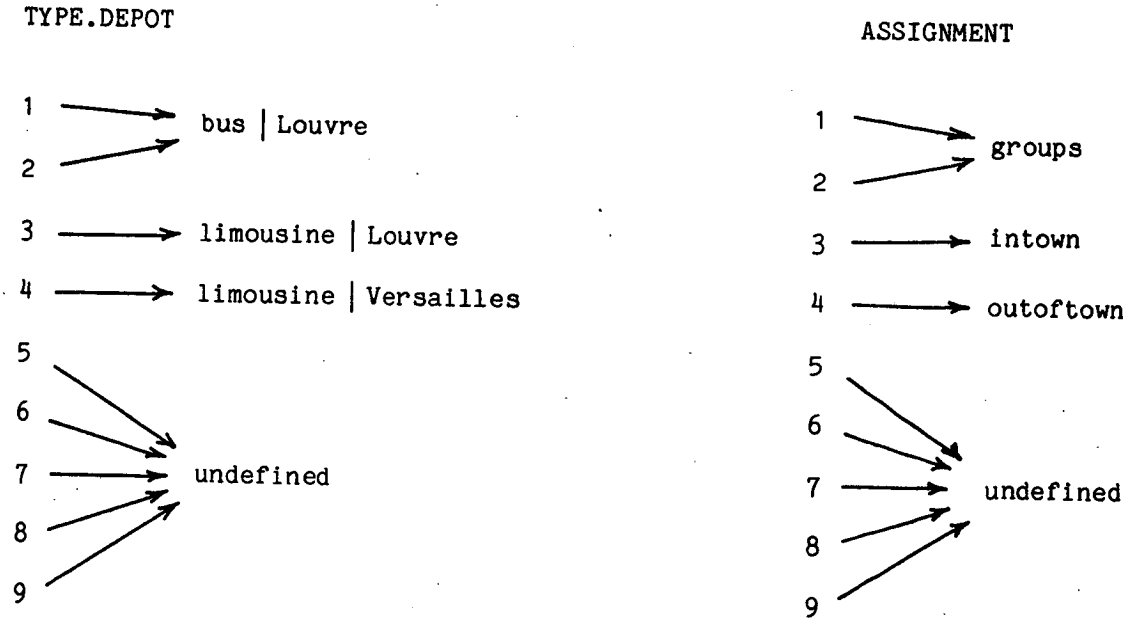


FIGURE 5. Two denotations of the product of T and D : using old names, or new names

Finally, we would like to remark that partitioning functions provide us with a notion of truth, if the empty set is called ~~false~~ and a nonempty set is called ~~true~~. Following this convention, a name is called false if its interpretation is the empty set and it is called true, otherwise. Thus, in Figure 1, the name "bus" is true, because its interpretation is the nonempty set {1,2}. Similarly, the name "bus | Louvre" in Figure 5 is true, because its interpretation is the nonempty set $\{1,2\} \cap \{1,2,3\}$. On the other hand the name "bus | Versailles" is false, because its interpretation is the set $\{1,2\} \cap \{4\}$ which is empty. We shall use frequently the terms ~~fact~~ and ~~data~~ instead of the term "name", and so we shall talk of true and false facts, or of true and false data.

The attribute ASSIGNMENT that we have defined earlier is "finer" than either TYPE or DEPOT in the following sense : if two objects of ε have the same ASSIGNMENT then they have the same TYPE ~~and~~ the same DEPOT. Suppose now that we would like to define a new attribute, called CONDITION, which is "coarser" than either RANK or SALARY (see Figure 1), in the following sense : if two objects of ε have the same RANK ~~or~~ the same SALARY then they have the same CONDITION. Consider, for example, the object 5 (Figure 2). This object has the same CONDITION as object 6, because 5 and 6 have the same SALARY. Similarly, 6 has the same CONDITION as 7, because 6 and 7 have the same RANK . Thus 5, 6, and 7 have the same CONDITION. On the other hand, 7 and 8 have neither the same RANK nor the same SALARY, so 7 and 8 do not have the same CONDITION. These examples motivate the following definitions (refer to Figure 2). Two objects x and y are s-equivalent with respect to R and S, denoted by $x \overset{s}{=} y$, if there is a sequence of objects $x=x_0, x_1, \ldots, x_n=y$, such that, for $i=0,1,\ldots,n-1$, either $R(x_i) = R(x_{i+1})$ ~~or~~ $S(x_i) = S(x_{i+1})$. The ~~sum~~ of R and S, denoted by R+S, is defined as follows :

$$\begin{aligned} \forall\, x \in \varepsilon \quad (R+S)(x) &= \emptyset \text{ , if } R(x) = \emptyset \text{ and } S(x) = \emptyset \\ &= \{y \mid x \overset{s}{=} y\} \text{ , otherwise} \end{aligned} \qquad (3)$$

It follows from this definition that if $(R+S)(x) = \{y_1,\ldots,y_n\}$ then

$$(R+S)(x) = R(y_1) \cup \ldots \cup R(y_n) \cup S(y_1) \cup \ldots \cup S(y_n) \qquad (4)$$

For example, referring to Figure 2, we find

$(R+S)(5) = \{5,6,7\}$, following (3)

$= \{5\} \cup \{6,7\} \cup \{6,7\} \cup \{5,6\} \cup \{5,6\} \cup \{7\}$, following (4).

These two equivalent representations of (R+S)(x) give rise to two equivalent representations of the sum of R and S, as shown in Figure 6. The first representation, denoted by R+S, shows the sum before computing unions. The second representation, denoted by C, shows the sum after computing unions. Of course R+S = C, but there is an important difference between the two representations when it comes to naming them (as was the case for the product). Indeed, we can name R+S using the identifiers that appear in Figure 1, and a naming convention. In fact, we only need to generalize the naming convention that we have used for naming the product :

> if μ is a naming of R and ν is a naming of S then the name of the set $R(y_1) \cup \ldots \cup R(y_n) \cup S(y_1) \cup \ldots \cup S(y_n)$ is the identifier $\{\mu(R(y_1)),\ldots,\mu(R(y_n))\} \mid \{\nu(S(y_1)),\ldots,\nu(S(y_n))\}$

Consider as an example the image (R+S)(5), seen earlier. We have :

$$\begin{aligned}(R+S)(5) &= \{5,6,7\} \\ &= R(5) \cup R(6) \cup R(7) \cup S(5) \cup S(6) \cup S(7) \\ &= \{5\} \cup \{6,7\} \cup \{6,7\} \cup \{5,6\} \cup \{5,6\} \cup \{7\}\end{aligned}$$

Using the namings of R and S from figures 1 and 2, and the above naming convention, we obtain the following name for the image (R+S)(5) :

{clerk,guide} | {high,average}

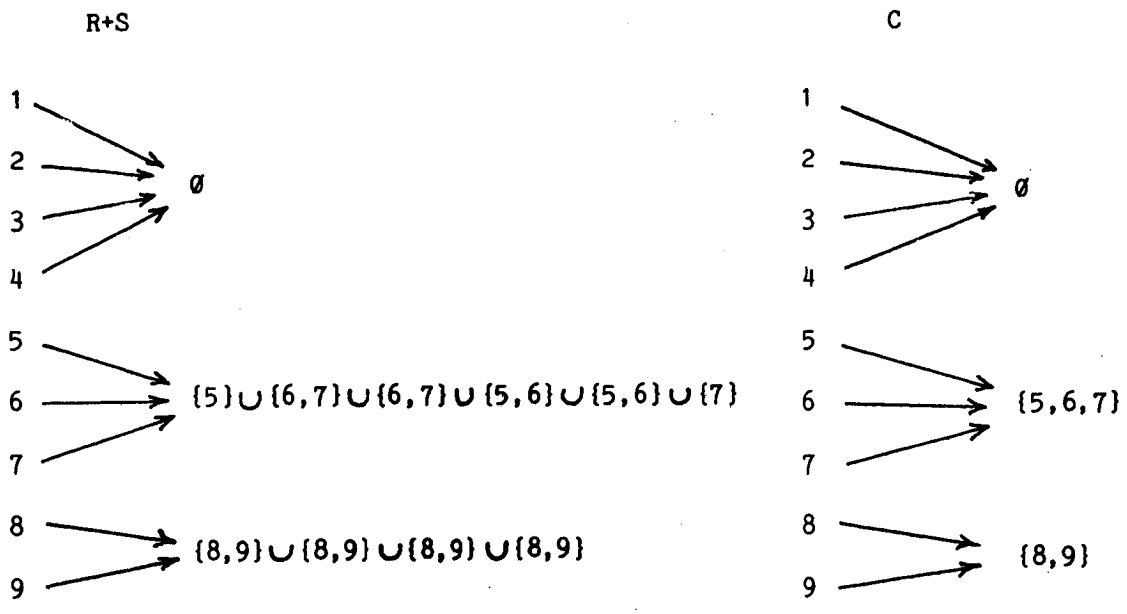


FIGURE 6. The sum of R and S before and after computing unions (denoted R+S and C, respectively)

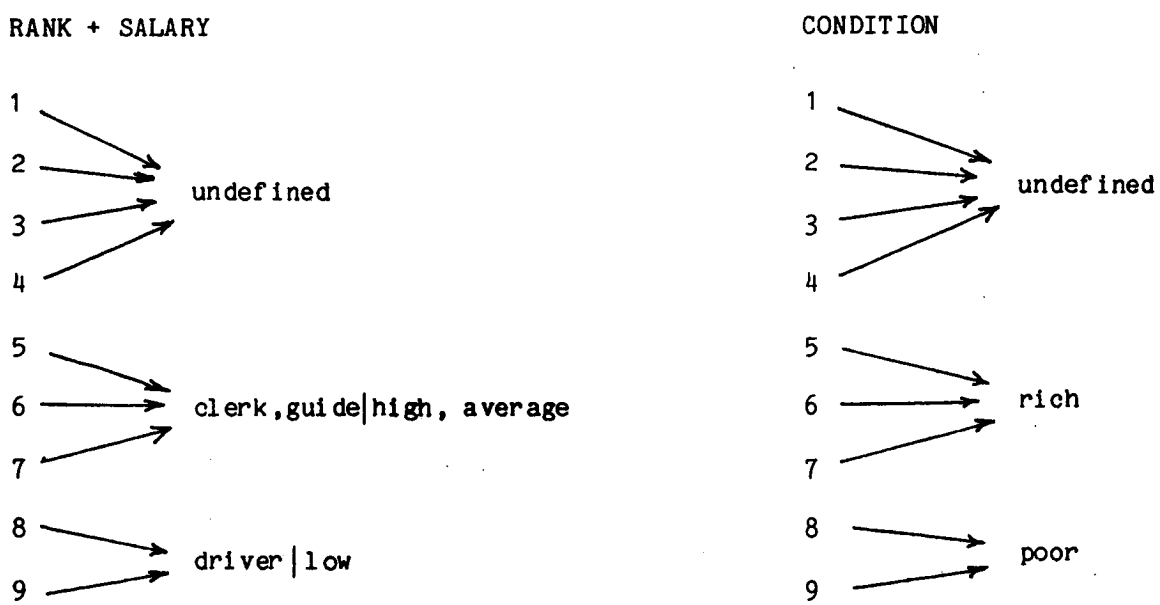


FIGURE 7. Two denotations of the sum of R and S : using old names, or new names

Proceeding in this manner we can name all the unions appearing in the range of R+S. The resulting names are shown in Figure 7, and they constitute the range of the function RANK+SALARY. (For simplicity, we have dropped the curly brackets). Thus we have been able to name the function R+S using old names and a naming convention. However, this is not possible for the function C as the nonempty sets appearing in its range have not been previously named. Thus, if we want to name the function C we must introduce new names for the nonempty sets {5,6,7} and {8,9} that appear in its range. Assume the following naming

{5,6,7} ⟷ rich
{8,9} ⟷ poor

Using this naming and the function C we obtain the function CONDITION shown in Figure 7.

Some important remarks, similar to those for products, are in order at this point. First, we remark that the sum R+S is also a partitioning, as it satisfies the properties (a) and (b) mentioned earlier. Second, although the functions R+S and C are equal, their named representations RANK+SALARY and CONDITION are not. That is, R+S or C carry the essential information, whereas RANK+SALARY and CONDITION are two different name-dependent representations of that information. Third, the function C carries "coarser" information than the function R in the sense that, if two objects of ε are associated with the same set of objects under R then they are associated with the same set of objects under C (that is, $R \leq C$). Finally, the interpretation of a fact such that {clerk,guide}|{high,average} as implied by the definition of the sum of R and S, is as follows : the facts "clerk", "guide", "high", and "average are all true and there is an ordering, say a_1, a_2, a_3, a_4, such that the facts $a_1|a_2$, $a_2|a_3$, $a_3|a_4$ are all true. As we can see from figures 1 and 2, this ordering (in our example) is the following : a_1=clerk, a_2=high, a_3=guide, and a_4=average. Indeed, the facts "clerk|high", high|guide", and "guide|average" are all true. Very roughly speaking, the above interpretation can be put into words as follows :

there are clerks and guides and they have high and average salaries

This kind of ambiguous statements constitute frequently the main sources of information in the enterprise.

Let us return once more to our example of a tourist agency. Suppose that some information, concerning the automobiles owned by the agency and the persons employed by the agency, needs to be recorded for daily use. Moreover, suppose that the attributes TYPE, DEPOT, RANK, and SALARY, are atomic attributes, in the sense that any information about the objects of interest to the enterprise will have to be expressed in terms of those attributes. Let us suppose that the attributes that are currently of interest to the enterprise, are the following composite attributes : TYPE.DEPOT, RANK.SALARY, and RANK+SALARY. Assume that the information that needs to be recorded consists of all facts of the attributes of interest that are currently true. This information is obtained by observing (true) facts in the enterprise and recording them in a storage medium. The set of all recorded facts of the attributes of interest is what we call current database. The set of attributes that are designated as atomic is what we call the database universe. In Figure 8 we see an example of a current database. The convention that we use in such a pictorial representation of a database is the following : for each attribute of interest Q, write down Q and, under Q, list the observed (true) facts of Q. In this way, the current database can be viewed as a function, say d_ε, which associates each attribute of interest Q with a set of facts of Q, say $d_\varepsilon(Q)$. Let us emphasize that the definition of the function d_ε depends on the conditions under which facts are observed and recorded. For example, the person(s) collecting information in the enterprise may fail to record some facts that are currently true. Thus, two different sets of conditions may produce two different current databases.

RANK.SALARY	
clerk	high
guide	average
driver	low

RANK+SALARY	
clerk,guide	high,average
driver	low

FIGURE 8. - A current database of the tourist agency.

The database model introduced in this paper is based on a single concept, that of a partitioning function. Such a function goes from the set ω of nonnegative integers into the set of all subsets of ω, and satisfies the postulates that we saw earlier in this section. The important property of a partitioning function is that, given a naming, it is transformed to an attribute. In Section 2, we give the formal definition of our model. The main tool is the lattice of partitionings. A database is defined as a function on a finite sublattice. Section 3 shows how new facts may be derived from facts that were explicitly introduced in the database. The deductive process uses two sources of information. First, assumptions on the conditions under which facts are observed and recorded in the database. Second, dependencies among facts, expressed as lattice equations. Section 4 has to do with deductive query answering in an important class of databases, called conjunctive databases. Section 5 shows how the relational model and various semantic models can be embedded into our model. Section 6 contains some concluding remarks and suggestions for further work.

2. THE FORMAL DEFINITION OF THE MODEL

In this section we give the formal definition of our model. The basic building block is a function called a partitioning. The main tool that we use is the lattice of partitionings. In Section 2.1 we give the formal definition of a partitioning and we study the lattice of partitionings. In Section 2.2 we show how databases can be seen as functions on finite sublattices of the lattice of partitionings.

2.1. THE LATTICE OF PARTITIONINGS

Let ω be the set of all objects that can possibly be of interest to any human enterprise. We shall refer to ω as the world. We assume ω to be a countable set, hence we can think of ω as being the set of nonnegative integers. The main innovation in this report is to model properties of the objects in ω by some special functions, mapping ω into the set $P\omega = \{ \tau \mid \tau \subseteq \omega\}$, the set of all subsets of ω.

Definition 1. A partitioning is a function A from ω into $P\omega$ such that :

(i) $\forall x \in \omega \quad A(x) \neq \emptyset \Rightarrow x \in A(x)$

(ii) $\forall x \in \omega \quad \forall y \in \omega \quad A(x) \neq A(y) \Rightarrow A(x) \cap A(y) = \emptyset$ □

The domain of a partitioning A, denoted by $\delta(A)$, is defined as follows :

$$\delta(A) = \{x \in \omega \mid A(x) \neq \emptyset\}$$

When we define a partitioning A, we only give the images A(x) such that x is in the domain of A. In the following example we give the definition of a partitioning A :

$$A(2) = \{2,3,5\},\ A(3) = \{2,3,5\},\ A(7) = \{7,8,9\},\ A(9) = \{7,8,9\}$$

The understanding here is that : $\delta(A) = \{2,3,7,9\}$. Therefore, $A(0)=\emptyset$, $A(1)=\emptyset$, $A(4)=\emptyset$, etc. The partitioning A, just defined, is shown pictorially in Figure 9. Two other partitionings, B and C, are shown as well. Let us note that the partitioning C satisfies the following property :

$$\forall x \in \omega \quad \forall y \in \omega \quad (C(x)\neq\emptyset \text{ and } y \in C(x)) \Rightarrow C(y)\neq\emptyset$$

As a consequence, the range of C is a partition of its domain. Let us note that Definition 1 does not require that a partitioning satisfy this property. For example, the partitioning B of Figure 9 does not satisfy this property. Indeed, we have :

$B(3)\neq\emptyset$ and $2 \in B(3)$ and $B(2)=\emptyset$

We shall come back to this remark shortly.

The range of a partitioning A, denoted by $\rho(A)$, is the image of the domain $\delta(A)$, that is,

$$\rho(A) = \{A(x) \mid x \in \delta(A)\}$$

Thus in Figure 9 the range of A consists of the sets {2,3,5} and {7,8,9}. It follows from Definition 1(ii) that if the range of a partitioning A is nonempty then it contains nonempty and mutually disjoint subsets of ω. A partitioning is called finite if its range is a finite set. For example, the partitionings of Figure 9 are all finite partitionings. Let us note that the domain of a finite partitioning is not necessarily a finite set. For example, consider the partitioning A defined as follows :

$$\forall\, x \in \omega \quad A(x) = \{y \in \omega \mid x \bmod 2 = y \bmod 2\}$$

The range of A contains only two sets, namely the set of even integers and the set of odd integers. Therefore A is a finite partitioning. Yet, the domain of A is the set ω which is infinite.

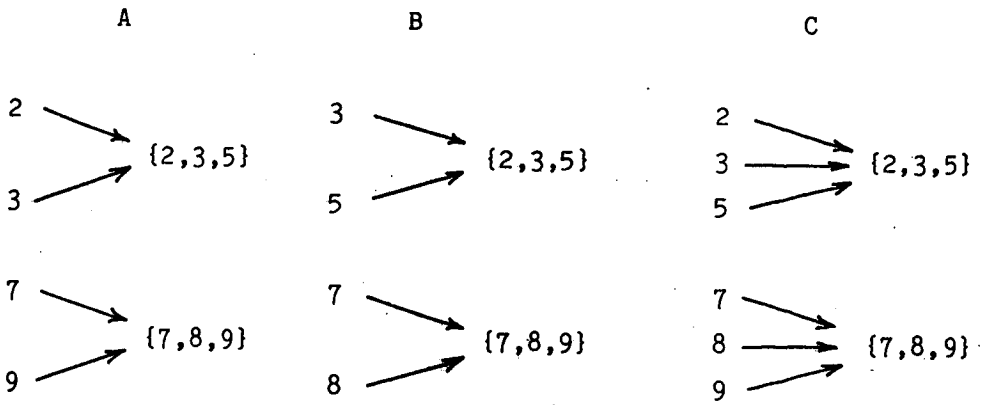


FIGURE 9. - Three partitionings with domains : $\delta(A)=\{2,3,7,9\}$, $\delta(B)=\{3,5,7,8\}$, and $\delta(C)=\{2,3,5,7,8,9\}$

Two special partitiongs, defined below, play a central role in our model.

Definition 2. The bottom and the top partitioning, denoted by ϕ and 1, respectively, are defined as follows :

$$\forall\, x \in \omega \quad \phi(x)=\emptyset \quad \text{and} \quad 1(x)=\omega \qquad \square$$

The justification of the terms "bottom" and "top" will be given shortly. For now, let us note the following :

$$\delta(\phi)=\emptyset, \quad \rho(\phi)=\emptyset, \quad \delta(1)=\omega, \quad \rho(1)=\{\omega\}$$

Partitionings whose range contains zero or one element, such as the partitionings ϕ and 1 just defined, are the most elementary and they deserve a name.

Definition 3. A partitioning f is called a fact or a datum, if its range contains at most one element. A fact f is called false if $f=\phi$, and true if $f\neq\phi$. Given a fact f and a partitioning A we say that f is a fact of A if

$$\forall\, x \in \omega \quad f(x)\neq\emptyset \Rightarrow f(x)=A(x) \qquad \square$$

For example, consider a partitioning f defined as follows :

$$f(2) = f(3) = f(5) = \{2,3,5\}$$

The range of f contains a single element, namely $\rho(f)=\{\{2,3,5\}\}$. It follows that f is a fact. Moreover, the fact f is true because $f \neq \phi$. Referring to Figure 9, it is easy to see that f is a fact of C. However, f is neither a fact of A nor a fact of B. In the opposite direction, it is easy to see that every fact of A and every fact of B is a fact of C.

Returning to Figure 9, let us note that, although the domains of A, B and C are different, their ranges are the same. We call two partitionings A and B name-equivalent if their ranges are equal that is, if $\rho(A)=\rho(B)$. This term is justified by the fact that, in reality,

what we store in a database are names of sets in the ranges of partitionings. Individual objects in ω associated with the same name cannot be distinguished on the basis of this name. Now, let $\Pi\omega$ be the set of all partitionings. It is easy to see that name-equivalence is an equivalence relation on the set $\Pi\omega$. Thus name-equivalence classifies partitionings into equivalence classes. The equivalence class of a partitioning A, denoted by [A], is defined as follows :

$$[A] = \{B \in \Pi\omega \mid \rho(A)=\rho(B)\}$$

For example, all three partitionings of Figure 9 belong to the same equivalence class, as their ranges are equal. Of course, as far as names are concerned, a single designated representative of each equivalence class would be sufficient for our purposes. Let us note that, for each partitioning A, there is always a partitioning $\hat{A}$ in [A] such that :

$$\forall\, x\in\omega \quad \forall\, y\in\omega \quad (\hat{A}(x)\neq\emptyset \ \underline{\text{and}} \ y\in\hat{A}(x)) \Rightarrow \hat{A}(y)\neq\emptyset \qquad (1)$$

In the example of Figure 9 $\hat{A}$ is the partitioning C. Let us note that the partitioning $\hat{A}$ enjoys two interesting properties. First, it follows from (1) that, for every B in [A], if f is a fact of B then f is also a fact of $\hat{A}$. Thus $\hat{A}$ carries all the relevant information of the equivalence class of A. Second, it follows from Definition 1 that there is only one partitioning in [A] satisfying (1). That is, the partitioning $\hat{A}$ is unique. We designate this unique partitioning as <u>the</u> representative of the equivalence class and we dignify it with a name.

Definition 4. A <u>full partitioning</u> is a function A from ω into $P\omega$ such that :

(i) $\forall\, x\in\omega \quad A(x)\neq\emptyset \Rightarrow x\in A(x)$

(ii) $\forall\, x\in\omega \quad \forall\, y\in\omega \quad A(x)\neq A(y) \Rightarrow A(x) \cap A(y) = \emptyset$

(iii) $\forall\, x\in\omega \quad \forall\, y\in\omega \quad (A(x)\neq\emptyset \ \underline{\text{and}} \ y\in A(x)) \Rightarrow A(y)\neq\emptyset$ □

Let us note that the bottom ϕ and the top **1** are both full partitionings. Let us also note that Definition 4(iii) can be restated as follows :

$$\forall\, x \in \omega \quad A(x) \neq \emptyset \Rightarrow A(x) \subseteq \delta(A) \quad \underline{\text{or}} \quad \forall\, x \in \omega \quad A(x) \neq \emptyset \Rightarrow A^{-1}(A(x)) = A(x)$$

It follows that, for every x in ω, A(x) is the domain of a fact of A. The range of this fact is $\{A(x)\}$ if $A(x) \neq \emptyset$, and it is empty otherwise.

It follows from Definition 4 that the range of a full partitioning is a partition of its domain. Thus a full partitioning is completely defined if we are given its range. In fact, we can set up a one-one correspondence between full partitionings and certain families of subsets of ω. Let us call <u>disjoint family</u> any family of sets which either is empty or it contains nonempty and mutually disjoint subsets of ω. For example, the family $\{\{2,3,5\},\{7,8,9\}\}$ is a disjoint family. Note that the range of every partitioning is a disjoint family. Thus, if we let $F\omega$ be the set of all full partitionings and $J\omega$ be the set of all disjoint families, then we can set up the following one-one correspondence between $F\omega$ and $J\omega$:

- with every full partitioning A in $F\omega$ associate its range $\rho(A)$ in $J\omega$
- with every disjoint family ξ in $J\omega$ associate the full partitioning A_ξ defined as follows :

$$\delta(A_\xi) = \bigcup\{\sigma \mid \sigma \in \xi\}$$
$$\forall\, x \in \omega \quad A_\xi(x) = \tau, \text{ if } x \in \tau \text{ and } \tau \in \xi$$
$$= \emptyset, \text{ otherwise.}$$

Note that, in this correspondence, the bottom ϕ is associated with the empty family. An important consequence of the one-one correspondence between full partitionings and disjoint families, is that a full partitioning can be "denoted" by, or "identified" to, a disjoint family of sets in $P\omega$. If, moreover, every object x in ω is "denoted" by, or "identified" to, the singleton $\{x\}$ in $P\omega$, then everything can be modeled within the "universal" domain $P\omega$. However, in order to keep the presentation simple, we shall continue to use symbols, such as x or A, to denote objects of ω and full partitionings on ω. In fact, we

shall confuse symbols and the things denoted by the symbols. In doing so, the underlying assumption, throughout the paper is the following unique name assumption : different symbols denote different things unless explicitly stated otherwise. For example, let A and B be two partitionings of $F\omega$. When we write "A=B" we must interpret this as "the partitioning (denoted by) A and the partitioning (denoted by) B are the same partitioning". Returning to full partitionings, let us emphasize, once more, that full partitionings are sufficient for data modelling. Thus, from now on, we shall consider only full partitionings, and the term "partitioning" will stand for "full partitioning".

Every partitioning in the set $F\omega$ induces an equivalence relation on the set ω as follows : two objects x and y in ω are called equivalent with respect to A, denoted by $x \equiv y(A)$, if $A(x)=A(y)$. Using the equivalence relations on ω induced by the partitionings of $F\omega$, we can define the product and the sum of partitionings.

Definition 5. Let A and B be two partitionings in $F\omega$. Two objects x and y in ω are called p-equivalent with respect to A and B, denoted by $x \hat{=} y$, if $A(x)=A(y)$ and $B(x)=B(y)$. The product of A and B, denoted A.B, is defined as follows :

$$\forall\, x \in \omega \quad (A.B)(x) = \emptyset \text{ , if } A(x) = \emptyset \text{ or } B(x) = \emptyset$$
$$= \{y \in \omega \mid x \hat{=} y\} \text{ , otherwise} \qquad \square$$

It follows from Definition 4 that the product A.B is also in $F\omega$ and that its domain is the intersection of the domains of A and B. That is, $\delta(A.B) = \delta(A) \cap \delta(B)$. It follows from Definition 3 that if a_1 and a_2 are facts of A then $a_1.a_2=\phi$. The behavior of the bottom and top partitioning, with respect to product, is as follows :

$$\forall\, A \in F\omega \quad \phi.A=A.\phi=\phi \quad \text{and} \quad 1.A=A.1=A$$

This follows easily from definitions 2 and 5. An immediate consequence of Definition 5 is the following equivalent definition of the product of A and B :

$$\forall\, x \in \omega \quad (A.B)(x) = A(x) \cap B(x)$$

It follows that if a is a fact of A and if b is a fact of B then a.b is a fact of A.B. Conversely, every fact f of A.B can be written as f=a.b, where a is a fact of A and b is a fact of B. A first consequence is that every fact of A.B is "finitely representable" in terms of facts of A and B. A second consequence is that if all the facts of A and B are named then we can use their names in order to derive names for all the facts of A.B. For example, if the name "Bus" stands for the fact a of A and if the name "Louvre" stands for the fact b of B then we can use the name "Bus.Louvre" to stand for the fact a.b of A.B.

Definition 6. Let A and B be two partitionings in Fω. Two objects x and y in ω are called s-equivalent with respect to A and B, denoted by $x \overset{\vee}{=} y$, if there is a sequence of objects $x=x_0,x_1,\ldots,x_n=y$, such that, for $i=0,1,\ldots,n-1$, either $A(x_i)=A(x_{i+1})$ or $B(x_i)=B(x_{i+1})$. The sum of A and B, denoted by A+B, is defined as follows : for all x in ω,

$$(A+B)(x) = \emptyset, \text{ if } A(x)=\emptyset \text{ and } B(x)=\emptyset$$
$$= \{y \mid x \overset{\vee}{=} y\}, \text{ otherwise} \qquad \square$$

It follows from Definition 6 that the sum A+B is also in Fω and that its domain is the union of the domains of A and B. That is, $\delta(A+B)=\delta(A) \cup \delta(B)$. The behavior of the bottom and top partitioning with repect to sum is as follows : for all A in Fω,

$$\phi+A=A+\phi=A \quad \text{and} \quad 1+A=A+1=1$$

This follows easily from definitions 2 and 6. An immediate consequence of Definition 6 is that, if $A(x)\neq\emptyset$ and $B(x)\neq\emptyset$, then $x \overset{\vee}{=} y$ if there is a sequence of facts $f_0,f_1,\ldots,f_n$ of A or B such that : $x\in\delta(f_0)$, $y\in\delta(f_n)$, $f_i.f_{i+1} \neq \phi$, $i=0,1,\ldots,n-1$. It follows that for any two objects x' and y' in the set $\delta(f_0) \cup \ldots \cup \delta(f_n)$ we have : $x' \overset{\vee}{=} y'$. Let us define a chain, with respect to A and B, as follows :

(1) - ϕ is a chain with respect to A and B

(2) If f is a fact of A or B then f is a chain with respect to A and B

(3) Let CH be a chain with respect to A and B. Let g be a fact of A or B. If $g.CH \neq \phi$ then CH+g is a chain with respect to A and B

(4) Nothing else is a chain with respect to A and B.

A chain CH, with respect to A and B is called <u>maximal</u> if, for every fact g of A or B, not appearing in CH, we have : $g.CH=\phi$. It follows that the bottom ϕ is a maximal chain with respect to A and B. Let us note that, for every maximal chain $CH \neq \phi$ and for every x in $\delta(CH)$ we have : $\{y \mid x \overset{\vee}{\leq} y\} = \delta(CH)$. It follows from Definition 6 that the sum of A and B can be defined (equivalently) as follows : for all x in ω,

$(A+B)(x) = \emptyset$, if $A(x)=\emptyset$ and $B(x)=\emptyset$

$= \delta(CH)$, if CH is a maximal chain and $x \in \delta(CH)$

It follows that every maximal chain with respect to A and B (including the bottom ϕ) is a fact of A+B. Conversely, every fact f of A+B can be expressed as f=CH, where CH is a maximal chain with respect to A and B. It follows that if all the facts of A and B are named then we can use their names in order to derive names for all facts of A+B. For example, suppose that the names "Clerk" and "Guide" stand for the facts a_1 and a_2 of A, respectively, and that the names "High" and "Average" stand for the facts b_1 and b_2 of B, respectively. If $a_1+b_1+b_2+a_2$ is a maximal chain with respect to A and B, then we can use the name "Bus+High+Average+Guide" to stand for the fact $a_1+b_1+b_2+a_2$ of A+B. Finally, let us note that if A and B are infinite partitionings then a maximal chain may contain infinitely many facts. It follows that if f is a fact of A+B then f is not always "finitely representable" in terms of facts of A and B.

We have seen so far the definitions of the product and the sum of two partitionings. We have also seen that the set $F\omega$ is closed under product and sum. That is, if A and B are two partitionings in $F\omega$ then A.B and A+B are also partitionings in $F\omega$. Moreover, the product and the sum of partitionings satisfy the lattice postulates. This is stated formally in the following proposition whose proof is omitted, as it is an immediate consequence of the definitions.

Proposition 1. For all partitionings A,B, and C in Fω the following is true :

Idempotence : $A.A = A$ and $A+A = A$

Commutativity : $A.B=B.A$ and $A+B=B+A$

Associativity : $A.(B.C) = (A.B).C$ and $A+(B+C) = (A+B)+C$

Absorption : $A.(A+B) = A$ and $A+(A.B) = A$ □

It follows from this proposition that the set Fω is a lattice [4].

Some important remarks on notation are in order at this point. First, for notational convenience, we shall assume that the product has priority over the sum. Thus, for example, we shall write "A+B.C" instead of "A+(B.C)". Second, we shall often write a finite partitioning as the sum of its facts. Let us recall that a partitioning is finite if its range is a finite set. For example, let A be a finite partitioning and let a_1 and a_2 be the only facts of A. Then $a_1.a_2=\phi$ and Definition 6 implies that $A=a_1+a_2$. Moreover, as the sum is commutative, the order in which the facts appear in the sum is immaterial. Third, we shall often write the sum of two finite partitionings, say A and B, as the sum of the maximal chains with respect to A and B. For example, let a_1, a_2, and a_3 be the only distinct facts of A and let b_1, b_2, and b_3 be the only distinct facts of B. Assume that $a_1+a_2+b_1+b_2$ and a_3+b_3 are the two distinct maximal chains with respect to A and B. From what we have seen on chains earlier, it follows that

$$A+B = (a_1+a_2+b_1+b_2) + (a_3+b_3)$$

where parentheses are used to delimit maximal chains.

We have seen that the set Fω is a lattice under product and sum of partitionings. The lattice operations induce a partial ordering on the set Fω, defined as follows :

Definition 7. Let A and B be two partitionings of Fω. We say that A is <u>finer</u> than B (or that B is <u>coarser</u> than A), denoted by $A \leq B$, if $A.B=A$ (equivalently, if $A+B=B$). □

The following equivalent definition of the ordering is an immediate consequence of definitions 5 and 7 :

$$A \leq B \quad \text{iff} \quad \forall\, x \in \omega \quad A(x) \subseteq B(x)$$

It follows that if $A \leq B$ then the domain of A is a subset of the domain of B that is, $\delta(A) \subseteq \delta(B)$. With regards to greatest lower bounds (glb) and least upper bounds (lub) we have :

$$\text{glb}(A,B) = A.B \quad \text{and} \quad \text{lub}(A,B) = A+B$$

The behavior of the bottom and top partitioning, with respect to the lattice ordering, is as follows :

$$\forall\, A \in F\omega \quad \phi \leq A \leq 1$$

Thus ϕ is the first element of the lattice and 1 is the last element, a fact that justifies their names as "bottom" and "top", respectively. It follows that the lattice $F\omega$ is bounded. However, this lattice is not distributive, as the example of Figure 10 shows. Indeed, we can verify easily that $A.(B+C)=A$, whereas $A.B+A.C \neq A$. Therefore, $A.(B+C) \neq A.B+A.C$ and thus the lattice $F\omega$ is not distributive.

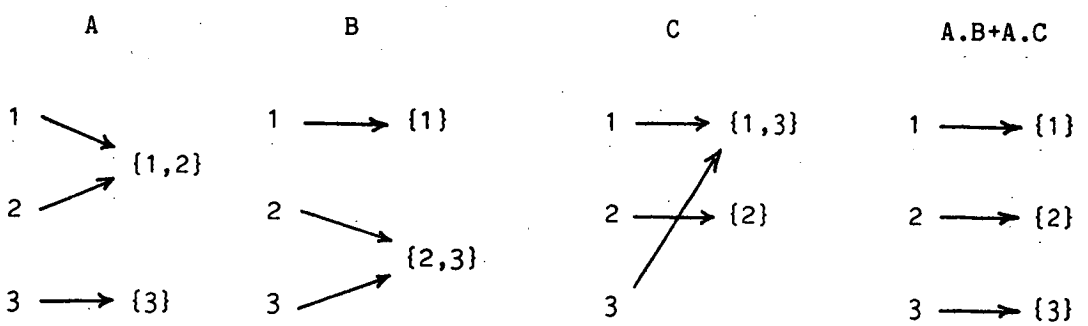


FIGURE 10. The lattice $F\omega$ is not distributive as $A.(B+C) \neq A.B+A.C$

Although the lattice $F\omega$ is not distributive, certain useful inequalities hold in it. They are summarized in the following proposition :

Proposition 2. Let A,B,C and D be partitionings of $F\omega$. Then we have :

(a) If $A \leq C$ and $B \leq D$ then $A.B \leq C.D$ and $A+B \leq C+D$

(b) $A.B+A.C \leq A.(B+C)$ and $A+B.C \leq (A+B).(A+C)$

(c) $A.B+B.C+C.A \leq (A+B)(B+C)(C+A)$

(d) $A.B+A.C \leq A.(B+A.C)$ □

Proof :

(a) It follows immediately from the definitions of product and sum

(b) Since $A \leq A$ and $B \leq B+C$, it follows from (a) that $A.B \leq A.(B+C)$. Similarly, we obtain $A.C \leq A.(B+C)$. Then (a) implies that $A.B + A.C \leq A.(B+C)$. The second inequality can be proved by a similar argument.

(c) Since $A \leq A+B$, $B \leq B+C$, and $A \leq C+A$, it follows from (a) that $A.B \leq (A+B).(B+C).(C+A)$. By similar arguments we obtain $B.C \leq (A+B).(B+C).(C+A)$ and $C.A \leq (A+B).(B+C).(C+A)$, and then (c) follows from (a).

(d) Since $A.B \leq B \leq B+AC$ and $A.C \leq B+A.C$, it follows from (a) that $A.B+A.C \leq B+A.C$. On the other hand, since $A.B \leq A$ and $A.C \leq A$, it follows from (a) that $A.B+A.C \leq A$. A new application of (a) on these two results yields (d). □

We have seen so far how the set $F\omega$ of all (full) partitionings can be turned into a lattice. Let us recall that a partitioning A of $F\omega$ is uniquely determined given its range $\rho(A)$. However A is not uniquely determined given only its domain $\delta(A)$ unless the range of A contains at most one element. Let us recall that partitionings of $F\omega$ whose range contains at most one element are called facts. Thus, if f is a fact then f is uniquely determined given its domain. Indeed, if $\delta(f) = \emptyset$ then $f = \phi$ else $\rho(f) = \{\delta(f)\}$. Using this property of facts and the lattice operations we can derive many useful properties of facts. We give here a non-exhaustive list of examples. The proofs are omitted as they follow immediately from the definitions. Let a, b, and c be facts of $F\omega$. Then we have :

$a.b \neq \phi \Rightarrow (a \neq \phi \text{ and } b \neq \phi)$

$a.b \neq \phi \Leftrightarrow \delta(a) \cap \delta(b) \neq \emptyset$

$a \leq b \Leftrightarrow a.b = a \Leftrightarrow \delta(a) \subseteq \delta(b)$

$a.b = c \Leftrightarrow \delta(a) \cap \delta(b) = \delta(c)$

$a+b = c \Leftrightarrow \delta(a) \cup \delta(b) = \delta(c)$

$a+b = c \Rightarrow (a.c = a \text{ and } b.c = b)$

$a.(b+c) = a.b+a.c \text{ and } a+b.c = (a+b).(a+c)$

The last property implies that, if we restrict our attention to facts only, then distributivity holds. Now, if we assume that the facts a and b are facts of the partitionings A and B, respectively, then we have the following additional properties :

$(A=B \text{ and } a.b \neq \phi) \Rightarrow a = b$

$(A \leq B \text{ and } a.b \neq \phi) \Rightarrow a \leq b$

$(A.B = A \text{ and } a.b \neq \phi) \Rightarrow a.b = a$

$(A+B = B \text{ and } a.b \neq \phi) \Rightarrow a+b = b$

Let us emphasize again that the list of properties given here is by no means exhaustive. Nevertheless, it is sufficient in order to indicate how deduction is done in the lattice $F\omega$. Here is another example revealing the deductive nature of our model. Let A, B, and C be three partitionings of $F\omega$. Suppose we are told that the fact a.b of A.B and the fact b.c of B.C are both true, that is, $a.b \neq \phi$ and $b.c \neq \phi$. Based on this information alone, we cannot say whether the fact a.b.c of A.B.C is true. Suppose now we are told that A.B=B. As $a.b \neq \phi$, it follows that a.b=b and, therefore, a.b.c = b.c. As b.c is true we deduce that a.b.c is true too. Note that we can reach the same conclusion if we are told that B.C=B, instead of A.B=B. Thus we have :

$(a.b \neq \phi \text{ and } b.c \neq \phi \text{ and } (A.B=B \text{ or } B.C=B)) \Rightarrow a.b.c \neq \phi$

This kind of deductive reasoning is one of the basic features of our model.

Given a partitioning Q of the lattice $F\omega$, we shall be interested in "instances" of Q, defined as follows :

Definition 8. Let Q be a partitioning of Fω. A partitioning R of Fω is an instance of Q (or less defined than Q) if, for all x in ω,

$$R(x) \neq \emptyset \Rightarrow R(x) = Q(x). \qquad \square$$

In Figure 11 we see two partitionings R and Q such that R is an instance of Q. Intuitively, we think of the instance R as made up by some of the facts of Q. For example, think of the domain of Q as being a set of persons and suppose that Q gives the age of each person. Then the instance R of Q gives the age of some of the persons in the domain of Q. Another way to look at instances is to think of Q as a partitioning of interest to some enterprise. That is, the enterprise is interested in collecting facts of Q and recording them in some medium. Then we can think of the instance R as made up of the recorded facts of Q. Of course, one or more (true) facts of Q may not be recorded thus R is made up by some of the facts of Q, namely by those facts of Q that were actually "observed" during the fact-collecting process.

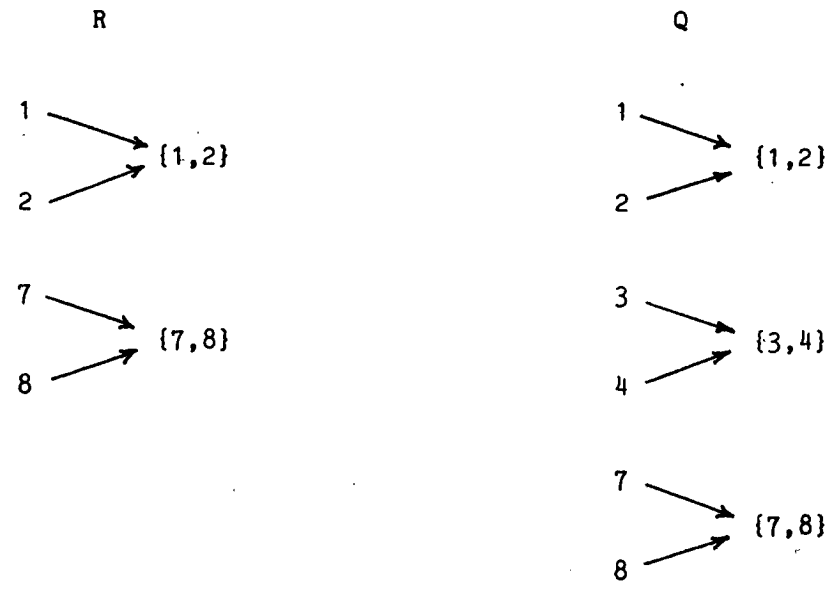


FIGURE 11. R is an instance of Q

It follows from Definition 8 that if R is an instance of Q then every fact of R is also a fact of Q. Moreover, $R \leq Q$ (trivially) and thus $\delta(R) \subseteq \delta(Q)$. Now, let R_1 and R_2 be two instances of Q. It follows from Definition 8 that :

$$\delta(R_1.R_2) = \delta(R_1) \cap \delta(R_2) \quad \underline{\text{and}} \quad \rho(R_1.R_2) = \rho(R_1) \cap \rho(R_2)$$
$$\delta(R_1+R_2) = \delta(R_1) \cup \delta(R_2) \quad \underline{\text{and}} \quad \rho(R_1+R_2) = \rho(R_1) \cup \rho(R_2)$$

The interesting point here is that the product and the sum act as intersection and union (respectively) not only on the domains but also on the ranges of R_1 and R_2. In fact we can even define the difference R_1-R_2 as follows :

Definition 9. Let Q be a partitioning in Fω. Let R_1 and R_2 be two instances of Q. The **difference** of R_1 and R_2, denoted by R_1-R_2, is defined as follows

$$\forall\, x \in \omega \quad (R_1-R_2)(x) = R_1(x), \text{ if } x \in (\delta(R_1)-\delta(R_2))$$
$$= \emptyset \text{ , otherwise}$$

□

It follows from this definition that the difference of two instances of Q is also an instance of Q. Furthermore, the difference of two instances of Q acts as set-theoretic difference on he domains and on the ranges of the two instances. More precisely, we have :

$$\delta(R_1-R_2) = \delta(R_1)-\delta(R_2) \quad \text{and} \quad \rho(R_1-R_2) = \rho(R_1)-\rho(R_2)$$

If R is an instance of Q then we refer to the difference Q-R as the complement of R with respect to Q. Given a partitioning Q of Fω, we denote by I(Q) the set of all instances of Q. Within the set I(Q) the product, the sum and the difference behave as set-theoretic intersection, union, and difference, respectively. Thus I(Q) is a distribututive lattice under product and sum of partitionings. We state this formally in the following proposition :

Proposition 3. Let Q be a partitioning of Fω. Let I(Q) be the set of all instances of Q. Then I(Q) is a distributive lattice under product and sum of partitionings.

□

In practice, the facts that are stored in a database are used in order to derive information about the objects of interest to the enterprise. Let ε denote the set of all objects of interest to the enterprise. This set is a subset of ω, the set of all objects that can possibly be of interest (to any enterprise). Now, every partitioning Q of $F\omega$ carries information about the objects of ε. This information can be found by restricting the function Q to the set ε, as follows :

Definition 10. Let Q be a partitioning in $F\omega$. Let ε be a subset of ω. The ε-instance of Q, denoted by Q_ε, is a partitioning defined as follows :

$$\forall\ x \in \omega \quad Q_\varepsilon(x) = Q(x), \text{ if } x \in \varepsilon$$
$$= \emptyset \text{ , otherwise} \qquad \square$$

Loosely speaking, Q_ε is a partitioning that "agrees" with the partitioning Q on the set ε, and it is "undefined" everywhere outside ε. Let us note that $Q_\omega = Q$. In general, the partitioning Q_ε is not a full partitioning. To see why, suppose that ε is the set of persons employeed by an enterprise, and that Q is a partitioning corresponding to "Marital Status". Suppose that the range of Q contains three subsets of the world ω, corresponding to "single", "married", and "other". Suppose also that the persons employed by the enterprise are either single or married. That is, suppose that the set ε contains only single or married persons. Now, in order for Q_ε to be a full partitioning, the set ε must contain <u>all</u> single persons and <u>all</u> married persons of the world ω, and this is not the case in general. If Q_ε is not a full partitioning, then we can replace it by its (unique) equivalent full partitioning. As we have explained earlier, this replacement makes no difference as far as facts are concerned. Thus, without loss of generality, we assume henceforth that Q_ε <u>is</u> a full partitioning that is, we assume that Q_ε is in $F\omega$. Under this assumption, it follows from Definition 10 that Q_ε is an instance of Q. The following proposition states an important property of ε-instances that we shall use shortly. The proof is omitted, as it follows immediately from Definition 10 and the definitions of product and sum.

Proposition 4. Let A and B be two partitionings in $F\omega$. Let ε be a subset of ω. Then $(A.B)_\varepsilon = A_\varepsilon.B_\varepsilon$ and $(A+B)_\varepsilon = A_\varepsilon+B_\varepsilon$ □

In conclusion, in this section, we have introduced the concept of partitioning and we have seen how the set $F\omega$ of all (full) partitionings can be turned into a lattice. We have also introduced the concept of a fact and we have shown that every finite partitioning can be written as the sum of facts. Finally, we have introduced the concept of an instance and that of an ε-instance. Roughly speaking, an instance of a partitioning Q is any restriction of Q and the ε-instance of Q is the restriction of Q to a specific set ε. We are now ready to define the concept of a database.

2.2. THE FORMAL DEFINITION OF A DATABASE

The basic building blocks in our definition of a database is what we call atomic partitionings, or simply atoms. These terms are justified by the fact that all other partitionings involved in the definition of a database are built-up from atoms using product and sum. What is required from each atom separately is that it carry some information about the objects in ω. That is, each atom must be different than the bottom ϕ. What is required from the atoms collectively is that they form a meaningful universe of discourse. That is, the product of all atoms must be different than the bottom ϕ. To explain this requirement, let us return once more to the example of a tourist agency that we discussed in the introduction. Referring back to Figure 1, recall that the attributes TYPE and DEPOT are undefined for employees, whereas RANK and SALARY are undefined for automobiles. Therefore it makes no sense to talk about, say, the DEPOT and the SALARY of an object, as there is no object for which both DEPOT and SALARY are defined. Thus the requirement that the product of all atoms be different than the bottom ϕ, corresponds to the following intuitive statement : "there must be an object of the world ω for which all atoms are defined". Of course, a set of atoms that does not satisfy this requirement can be always decomposed into subsets (eventually, singleton subsets) that do satisfy this requirement. For example, referring back to Figure 2, we note that the set {T,D,R,S} is not a meaningful universe, as T.D.R.S = ϕ. However, its subsets {T,D} and

{R,S} are meaningful universes, as T.D ≠ ϕ and R.S ≠ ϕ. We summarize our discussion so far in the following definition :

Definition 11. A universe is any finite nonempty subset U of Fω such that glb(U) ≠ ϕ □

As we have already mentioned, the elements of a universe are called atoms. Loosely speaking, the atoms are the basic attributes of the objects of interest to the enterprise. Composite attributes are built-up from atoms using product and sum. Thus, given a universe U, its closure under product and sum contains all attributes that can possibly be of interest to the enterprise.

Let U be a universe. The closure of U under product and sum is denoted by L(U). For example, if U = {A,B,C} then we have :

L(U) = {A, B, C, A.B, B.C, A.B.C, A+B, B+C, A+C, A+B+C, A+B.C, B+A.C, C+A.B, A.(B+C), B.(A+C), C.(A+B)}

As the universe U is a finite subset of Fω, the set L(U) is a finite sublattice of Fω. The partitionings of L(U) that are not in U are called composite partitionings, as they are built-up from atoms using product and sum. In our previous example, A+B.C is a composite partitioning. A partitioning Q of L(U) is called conjunctive if either it is an atom, or it is the product of atoms. Otherwise, Q is called disjunctive. For example, if U = {A,B,C} then the conjunctive partitionings of L(U) are the following : A, B, C, A.B, B.C, A.C, and A.B.C. All other partitionings of L(U) are disjunctive. A fact of an atomic partitioning is called an atomic fact. A fact of a composite partitioning is called a composite fact. A composite fact can be conjunctive or disjunctive. For example, if a and b are atomic facts then a.b is a composite fact which is conjunctive.

Let ε be the set of all objects of interest to the enterprise. This set is a subset of the world ω, the set of all objects that can possibly be of interest (to any enterprise). As we have explained in the introduction, the set ε changes during the life of the enterprise.

Nevertheless, here we are interested only in a specific moment in the life of the enterprise, at which moment the set of all objects of interest is ε. We use the term "current" to refer to that specific moment. Thus, ε is the set of all objects currently of interest to the enterprise. Now, every partitioning Q of L(U) carries information about the objects of ε. This information is represented by the ε-instance of Q, denoted by Q_ε (see Definition 10). In other words, Q_ε is the "part" of Q concerning ε. In principle, the ε-instances of <u>all</u> partitionings of L(U) are currently of interest to the enterprise. In reality, only the observed facts of some "convenient" partitionings are recorded. Of course, given a partitioning Q of L(U), the observed facts of Q are not necessarily <u>all</u> the facts of Q_ε. Some facts of Q_ε may not be observed and, therefore, may not be recorded. What is actually recorded in the database is an instance of Q_ε. Thus, if we let $I(Q_\varepsilon)$ denote the set of all instances of Q_ε, then we have the following definition of a database :

<u>Definition 12.</u> Let ε be a subset of ω. Let U be a universe. A <u>database on ε and U</u> is any function d_ε from the set L(U) into the set $F\omega$ such that : $\forall\, Q \in L(U)\quad d_\varepsilon(Q) \in I(Q_\varepsilon)$ □

If ε is the set of all objects currently of interest to the enterprise, then any given database on ε and U will be referred to as "the current database". Morevover, if d_ε is the current database then, for each Q in L(U), the instance $d_\varepsilon(Q)$ will be referred to as "the current database instance of Q". The <u>domain</u> of a database d_ε, denoted by $dom(d_\varepsilon)$, is defined as follows :

$$dom(d_\varepsilon) = \{Q \in L(U) \mid d_\varepsilon(Q) \neq \phi\}$$

When we define a database d_ε, we only give the instances $d_\varepsilon(Q)$ such that Q is in the domain of d_ε. For example, let U = {A,B,C}. Then the following is an example of database definition :

$$d_\varepsilon(A.B) = a_1.b_1 + a_2.b_2$$
$$d_\varepsilon(B.C) = b_1.c_1 + b_3.c_3$$
$$d_\varepsilon(B+C) = (b_1+b_2+c_1) + (b_3+c_3)$$

The understanding here is that : $dom(d_\varepsilon) = \{A.B, B.C, B+C\}$. The database just defined can be represented pictorially as shown in Figure 12. The convention that we use in order to derive such a pictorial representation of a database is the following : for each Q in the domain of d_ε, write down Q and, under Q, list the facts of $d_\varepsilon(Q)$. A database is called conjuntive if every partitioning in its domain is a conjunctive partitioning. Otherwise, the database is called disjunctive. For example, the database of Figure 12 is a disjunctive database.

A.B	B.C	B+C
$a_1.b_1$	$b_1.c_1$	$b_1+b_2+c_1$
$a_2.b_2$	$b_3.c_3$	$b_3+c_2+c_3$

FIGURE 12. A database on universe $U = \{A,B,C\}$

Two special databases, defined below, play an important role in our model.

Definition 13. The empty and the full database on ε and U, denoted by e_ε and f_ε, respectively, are defined as follows :

$$\forall\ Q \in L(U) \quad e_\varepsilon(Q) = \phi \quad \text{and} \quad f_\varepsilon(Q) = Q_\varepsilon \qquad \square$$

It follows from this definition that $dom(e_\varepsilon) = \phi$ and $dom(f_\varepsilon) = L(U)$. Let us recall that, in our discussions, the universe U and the set ε remain fixed. We denote by $D\varepsilon$ the set of all databases on ε and U. The lattice structure of partitionings induces a lattice structure on databases in a very natural and straightforward manner.

Definition 14. Let $D\varepsilon$ be the set of all databases on ε and U. Let d_ε and d'_ε be two databases in $D\varepsilon$. The product and the sum of d_ε and d'_ε, denoted by $d_\varepsilon.d'_\varepsilon$ and $d_\varepsilon+d'_\varepsilon$, respectively, are defined as follows :

$$\forall\ Q \in L(U) \quad (d_\varepsilon.d'_\varepsilon)(Q) = d_\varepsilon(Q).d'_\varepsilon(Q)$$
$$\forall\ Q \in L(U) \quad (d_\varepsilon+d'_\varepsilon)(Q) = d_\varepsilon(Q)+d'_\varepsilon(Q) \qquad \square$$

It follows from Definition 14 that the product $d_\varepsilon.d'_\varepsilon$ is also in $D\varepsilon$ and that its domain is the intersection of the domains of d_ε and d'_ε. That is, $dom(d_\varepsilon.d'_\varepsilon) = dom(d_\varepsilon) \cap dom(d'_\varepsilon)$. The behavior of the empty and the full database, with respect to product, is as follows :

$$\forall\, d_\varepsilon \in D\varepsilon \quad e_\varepsilon.d_\varepsilon = d_\varepsilon.e_\varepsilon = e_\varepsilon \text{ and } f_\varepsilon.d_\varepsilon = d_\varepsilon.f_\varepsilon = d_\varepsilon$$

Similarly, it follows from Definition 14, that the sum $d_\varepsilon+d'_\varepsilon$ is also in $D\varepsilon$ and that its domain is the union of the domains of d_ε and d'_ε. That is, $dom(d_\varepsilon+d'_\varepsilon) = dom(d_\varepsilon) \cup dom(d'_\varepsilon)$. The behavior of the empty and the full database, with respect to sum, is as follows :

$$\forall\, d_\varepsilon \in D\varepsilon \quad e_\varepsilon+d_\varepsilon = d_\varepsilon+e_\varepsilon = d_\varepsilon \text{ and } f_\varepsilon+d_\varepsilon = d_\varepsilon+f_\varepsilon = f_\varepsilon$$

It is easy to see that the product and the sum of databases satisfy the lattice postulates (see Proposition 1). It follows that the set $D\varepsilon$ is a lattice. The lattice ordering can be defined, using the lattice operations, as follows :

Definition 15. Let d_ε and d'_ε be two databases in $D\varepsilon$. We say that d_ε is less defined than d'_ε (or that d'_ε is more defined than d_ε), denoted by $d_\varepsilon \leq d'_\varepsilon$, if $d_\varepsilon = d_\varepsilon.d'_\varepsilon$ (equivalently, if $d'_\varepsilon = d'_\varepsilon + d_\varepsilon$) □

Roughly speaking, d_ε is less defined than d'_ε if all facts stored in d_ε are also stored in d'_ε (but d'_ε may also contain additional facts that are not contained in d_ε). The following equivalent definition of the ordering is an immediate consequence of definitions 14 and 15.

$$d_\varepsilon \leq d'_\varepsilon \quad \text{iff} \quad \forall\, Q \in L(U) \quad d_\varepsilon(Q) \leq d'_\varepsilon(Q)$$

It follows that if $d_\varepsilon \leq d'_\varepsilon$ then the domain of d_ε is a subset of the domain of d'_ε. That is, $dom(d_\varepsilon) \subseteq dom(d'_\varepsilon)$. With regards to greatest lower bounds (glb) and least upper bounds (lub), we have :

$$glb(d_\varepsilon, d'_\varepsilon) = d_\varepsilon.d'_\varepsilon \quad \text{and} \quad lub(d_\varepsilon, d'_\varepsilon) = d_\varepsilon+d'_\varepsilon$$

The behavior of the empty and the full database with respect to the lattice ordering is as follows :

$$\forall\, d_\varepsilon \in D\varepsilon \quad e_\varepsilon \leq d_\varepsilon \leq f_\varepsilon$$

Thus, e_ε is the first element in the lattice $D\varepsilon$ and f_ε is the last element. It follows that the lattice $D\varepsilon$ is bounded. The lattice $D\varepsilon$ is also distributive. This is an immediate consequence of Definition 14 and Proposition 3.

In conclusion, in this section, we have introduced the concept of a current database on a set of objects ε and a universe U. We have also seen how the set $D\varepsilon$, of all current databases, can be turned into a bounded lattice. The last element of this lattice, is the full database f_ε and it can be considered as the current database on ε and U. Any other database d_ε of $D\varepsilon$ is less defined than f_ε and can be considered as an "imperfect" specification of f_ε. Starting from a database d_ε we can "improve" it by deducing new facts from those explicitly stored in the database. This deductive process is the subject of the following section.

3. THE DEDUCTIVE PROCESS

A database represents our knowledge of the enterprise coming from the observation of individual facts. This knowledge can be "improved", if we use other sources of information in order to deduce new facts from those observed and recorded. The main sources of information are :

(1) The definition of the database model being used, referred to as model-oriented information, and

(2) The application being modeled, referred to as application-oriented information.

Model-oriented information is fixed, once the model is fixed. In this paper, model-oriented information is whatever we have seen in Section 2. For example, if A is a partitioning and if a_1 and a_2 are facts of A

then we know that $a_1 . a_2 = \phi$ (it follows from the unique name assumption). On the other hand, application-oriented information is not fixed but varies with the application being modeled. In this section we discuss two sources of application-oriented information :

(2a) Assumptions on the conditions under which individual facts are observed and recorded in the database, and

(2b) Information inherent in the facts being observed, such as "an employee is female or male but not both"

In the remaining of this paper, model-oriented information and application-oriented information are referred to, collectively, as external information.

3.1. ASSUMPTIONS

The first source of information is the set of assumptions that can reasonably be made on the current database d_ε. These assumptions are necessary because observation and recording of facts is a human activity susceptible to errors. The assumptions that we discuss in this section ensure that :

(1) All conjunctive facts that have been recorded in the database are currently true facts

(2) All atomic facts that are currently true have been recorded in the database

(3) All disjunctive facts are maximal chains

We introduce our assumptions using an example. Let U = {A,B} and consider a database d_ε defined as follows :

$$d_\varepsilon(A.B) = a_1 . b_1 + a_2 . b_2$$
$$d_\varepsilon(A+B) = (a_1 + b_1 + b_3) + (a_2 + b_2)$$

This database is shown pictorially in Figure 13. The conjunctive facts recorded in the database d_ε are the following : $a_1.b_1$, $a_2.b_2$, a_1, a_2, b_1, b_2, and b_3. Now, it is conceivable that some of these facts are false and that have been accidentally recorded in the database. However, by just looking at the database d_ε we have no means of knowing which facts are true and which are false. Hence the following assumption :

Assumption 1 : True Conjunctive Facts.
Let d_ε be a given database on universe U. For every conjunctive partitioning Q in L(U), if a fact q of Q is recorded in the database then q is a true fact of Q_ε □

If we adopt Assumption 1 then we know that all facts recorded in the given database d_ε are true. However, we have no information whatsoever about facts that are currently true but have probably not been recorded in the database. The following assumption says that, although some composite facts that are currently true may have not been recorded in the database, all atomic facts that are currently true have been recorded in the database.

Assumption 2 : Complete Atomic Facts.
Let d_ε be a given database on universe U. For all Q in U, if q is a true fact of Q_ε then q is recorded in the database d_ε □

The significance of this assumption can be better understood in conjunction with Assumption 1. Indeed, if we adopt assumptions 1 and 2, then the following statement is true : for all Q in U, q is a true fact of Q_ε iff q is recorded in the database d_ε. Thus, referring to our example of d_ε (see Figure 13), if we adopt assumptions 1 and 2 then we know that a_1 and a_2 are the only true facts of A_ε and that b_1,b_2 and b_3 are the only true facts of B_ε. It follows that the only facts of A.B that can possibly be true facts of $(A.B)_\varepsilon$ are : $a_1.b_1$, $a_1.b_2$, $a_1.b_3$, $a_2.b_1$, $a_2.b_2$, and $a_2.b_3$. Two of these facts, namely $a_1.b_1$ and $a_2.b_2$ are recorded in the database d_ε. It follows from Assumption 1 that $a_1.b_1$ and $a_2.b_2$ are true. However, the remaining four facts, namely $a_1.b_2$, $a_1.b_3$, $a_2.b_1$, and $a_2.b_3$ cannot be proved true or false based on assumptions 1 and 2 alone.

One further asumption is needed for disjunctive partitionings. To see why, refer again to our example of d_ε (see Figure 13) and consider the sum $a_1+b_1+b_3$ recorded there. This sum is supposed to be a fact of $(A+B)_\varepsilon$. That is, the sum $a_1+b_1+b_3$ is supposed to be a maximal chain with respect to A and B. However, it is conceivable that, while the maximal chain is actually the sum $a_1+b_1+b_3+b_4$, only part of it has been observed and recorded, namely the sum $a_1+b_1+b_3$. Hence the following assumption :

Assumption 3 : Maximal Chains.
Let d_ε be a given database on universe U. For every disjunctive partitioning Q in L(U), $d_\varepsilon(Q)$ is given as a sum of maximal chains □

Let us recall that maximal chains in a sum are delimited by parentheses. If we adopt Assumption 3 in our example of d_ε (Figure 13) then we know that the sums $a_1+b_1+b_3$ and a_2+b_2 are maximal chains with respect to A_ε and B_ε. It is important to note what happens if we do not adopt Assumption 3. Indeed, two problems arise in that case :

(i) It is conceivable that the sums $a_1+b_1+b_3$ and a_2+b_2 may not even be chains

(ii) Even if we assume that $a_1+b_1+b_3$ and a_2+b_2 are different chains, we have no means of knowing whether they are parts of the same maximal chain (unless we adopt Assumption 2 in conjunction with Assumption 3).

In the remaining of this paper we adopt assumptions 1 and 2 when we discuss conjunctive databases, and assumptions 1,2, and 3 when we discuss disjunctive databases. In practice, however, the particular set of assumptions to be adopted in an application is determined by the conditions under which facts are observed and recorded in the database. For instance, we may decide not to adopt Assumption 2, in a specific application, if we are not sure that all currently true facts have been recorded. No matter which assumptions are adopted, it is important to remember that assumptions can be used to deduce new facts from those that were explicitly introduced in the

database. This is shown pictorially in Figure 13, where starting from a database d_ε and assumptions 1,2, and 3, we have derived a new database d'_ε as follows :

(a) the conjunctive facts a_1, a_2, b_1, b_2, and b_3 are recorded in the database d_ε. It follows from Assumption 1 that they are all true. Therefore, $d'_\varepsilon(A) = a_1 + a_2$ and $d'_\varepsilon(B) = b_1 + b_2 + b_3$

(b) the conjunctive facts $a_1.b_1$ and $a_2.b_2$ are recorded in d_ε. It follows from Assumption 1 that they are true. The unique name asumption (from Section 2.1) implies that b_1 and b_3 are different facts of B. It follows that $b_1.b_3 = \phi$. It follows from Assumption 3 that the sum $a_1 + b_1 + b_3$ of $d_\varepsilon(A+B)$ is a maximal chain. As $b_1.b_3 = \phi$, it follows (from the definition of a chain) that $a_1.b_3 \neq \phi$. Therefore, $d'_\varepsilon(A.B) = a_1.b_1 + a_2.b_2 + a_1.b_3$

(c) it follows from Assumption 3 that $d'_\varepsilon(A+B) = d_\varepsilon(A+B)$.

It is important to note that the facts recorded in d'_ε are all those facts that can be proved true using d_ε and assumptions 1, 2, and 3. It follows that $d_\varepsilon \leq d'_\varepsilon$.

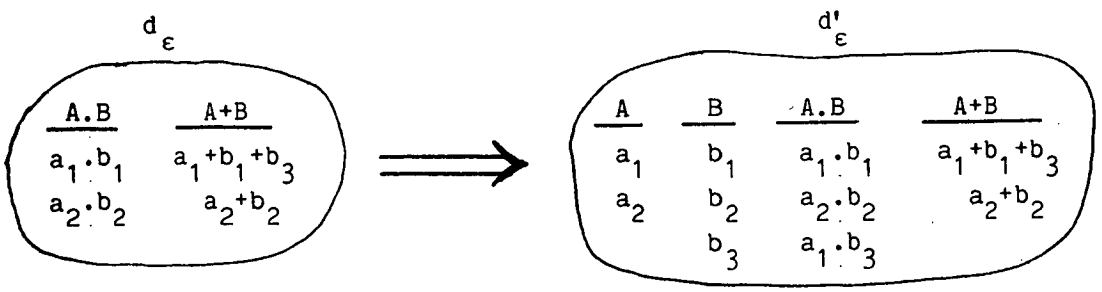


FIGURE 13. The given database d_ε and assumtions 1,2, and 3 imply a new database d'_ε.

3.2. DEPENDENCIES

Observation and recording of current facts of the enterprise results in a database, say d_ε. This database belongs to $D\varepsilon$, the set of all databases on ε, and represents our knowledge of the enterprise coming from the observation of individual facts. In a sense, the

database d_ε is our starting point. Using the assumptions adopted we can improve our knowledge of the enterprise in the sense that, starting from d_ε, we can derive a new database d'_ε such that $d_\varepsilon \leq d'_\varepsilon$. Our knowledge of the enterprise can be further improved based on information inherent in the facts being observed. Examples of such information are the following :

an employee is female or male but not both
an employee can work in only one department

This kind of information is expressed in our model in the form of equations called dependencies.

Definition 16. Let U be a universe. A dependency on U is any equation of the form q=r such that q and r are instances of partitionings of L(U) □

For example, if U = {A,B,C} then the equations A.B=C and $A_\varepsilon = B_\varepsilon + C_\varepsilon$ are dependencies on U. Similarly, if a,b, and c are facts of A,B, and C, respectively, the the equation a = b+c is a dependency on U. Let us recall (from Section 2) that an equation such as A.B=C must be interpreted as follows : the partitioning denoted by A.B and the partitioning denoted by C are the same partitioning. Let us also recall that, for all Q in L(U), the following equations are true : Q=Q.Q and Q=Q+Q. We shall refer to such equations as trivial dependencies. It is important to understand how an information such as "an employee has one and only one salary" can be represented by an equation. So, let us consider the universe U = {E,S}, where E stands for "employee" and S stands for "salary". Let j be a fact of E and let f be a fact of S, where j stands for "John" and f stands for "forty thousand". Now, suppose that the fact j.f is true that is, suppose that "John earns forty thousand" is true. Then we have :

$$(E.S = E \text{ and } j.f \neq \phi) \Rightarrow j \leq f$$

It follows that for every fact t of S we have : $t \neq f \Rightarrow j.t = \phi$. For example, if t stands for "thirty thousand" then "John earns thirty thousand" is false.

From our previous example of employees and salaries it is clear how dependencies are used in the deductive process. Here is another example of deduction, using dependencies. Let U = {A,B,C} and consider a database d_ε on U defined as follows :

$$d_\varepsilon(A.B) = a_1.b_1$$
$$d_\varepsilon(B+C) = (b_1+c_1+c_2)$$

Let us recall that we use parentheses in order to delimit maximal chains. Thus, in the above definition of d_ε, the instance $d_\varepsilon(B+C)$ consists of a single maximal chain, namely $b_1+c_1+c_2$. As the database d_ε just defined is disjunctive, we shall adopt assumptions 1,2, and 3. Moreover, suppose that we are given some general information in the form of the following equation : A=B+C. Using the database d_ε, the assumptions adopted, and the given equation, we can derive a new database d'_ε as shown in Figure 14. Here are some examples of how the facts of d'_ε are deduced from the facts of d_ε :

A,B, and C : It follows from Assumption 1 that the conjunctive facts a_1, b_1, c_1, and c_2 are all true. Therefore $d'_\varepsilon(A) = a_1$, $d'_\varepsilon(B) = b_1$, and $d'_\varepsilon(C) = c_1+c_2$, as shown in Figure 14.

A.B and B.C : It follows from Assumption 1 that the conjunctive fact $a_1.b_1$ is true. Therefore, $d'_\varepsilon(A.B) = a_1.b_1$. It follows from Assumption 3 that the sum $b_1+c_1+c_2$ of d_ε is a chain. Therefore the facts $b_1.c_1$ and $b_1.c_2$ are true (from the definition of a chain). It follows that $d'_\varepsilon(B.C) = b_1.c_1+b_1.c_2$, as shown in Figure 14.

A.C and A.B.C : As $a_1.b_1$ is true, it follows that $a_1.(b_1+c_1+c_2)$ is true too. As A=B+C, it follows that $a_1=b_1+c_1+c_2$ that is,

$$(A=B+C \text{ and } a_1.(b_1+c_1+c_2) \neq \phi) \Rightarrow a_1 = b_1+c_1+c_2$$

It follows that $a_1.c_1 \neq \phi$ and $a_1.c_2 \neq \phi$. Therefore, $d'_\varepsilon(A.C) = a_1.c_1+a_1.c_2$. On the other hand, as $a_1=b_1+c_1+c_2$ and $b_1.c_1 \neq \phi$, it follows that $a_1.b_1.c_1 \neq \phi$. Similarly, we can deduce that $a_1.b_1.c_2 \neq \phi$. That is ,

$(a_1=b_1+c_1+c_1$ and $b_1.c_1 \neq \phi) \Rightarrow a_1.b_1.c_1 \neq \phi$

$(a_1=b_1+c_1+c_2$ and $b_1.c_2 \neq \phi) \Rightarrow a_1.b_1.c_2 \neq \phi$

Therefore $d'_\varepsilon(A.B.C) = a_1.b_1.c_1 + a_1.b_1.c_2$, as shown in Figure 14.

A+B : It follows for assumptions 1 and 2 that a_1 is the only true fact of A_ε and that b_1 is the only true fact of B_ε. As the fact $a_1.b_1$ is true, it follows that a_1+b_1 is a maximal chain with respect to A_ε and B_ε. Therefore $d'_\varepsilon(A+B) = (a_1+b_1)$, as shown in Figure 14.

A+B.C : It follows from assumptions 1 and 2 that a_1 is the only true fact of A_ε, and that $b_1.c_1$ and $b_1.c_2$ are the only true facts of $(B.C)_\varepsilon$. On the other hand, the facts $a_1.b_1.c_1$ and $a_1.b_1.c_1$ are true. It follows that $a_1+b_1.c_1+b_1.c_2$ is a maximal chain with respect to A_ε and $(B.C)_\varepsilon$. Therefore $d'_\varepsilon(A+B.C) = a_1+b_1.c_1+b_1.c_2$, as shown in Figure 14.

A.(B+C) : We have already seen that $a_1=b_1+c_1+c_2$. Therefore the fact $a_1.(b_1+c_1+c_2)$ is true. It follows that $d'_\varepsilon(A.(B+C))=a_1.(b_1+c_1+c_2)$.

In the light of the previous examples, the reader should be able to follow through the remaining deduction steps that produce the database d'_ε of Figure 14. It is evident, however, that what we need here is a procedure for proving facts true or false, based on the given database d_ε and the external information I. For such a procedure the reader is referred to [3].

3.3. CONSISTENCY

We have seen so far that, given a database d_ε and external information I, we can derive a new database d'_ε such that $d_\varepsilon \leq d'_\varepsilon$. The database d'_ε consists of all facts that we can prove true based on d_ε and I. Thus d_ε and I are the premises of our deduction and, as such, they must be consistent. That is to say, there must be no contradiction between d_ε and I.

Definition 17. We say that a database d_ε is consistent with external information I (or that d_ε satisfies I), denoted by $d_\varepsilon \models I$, if we can define the atomic facts of d_ε so that I is verified □

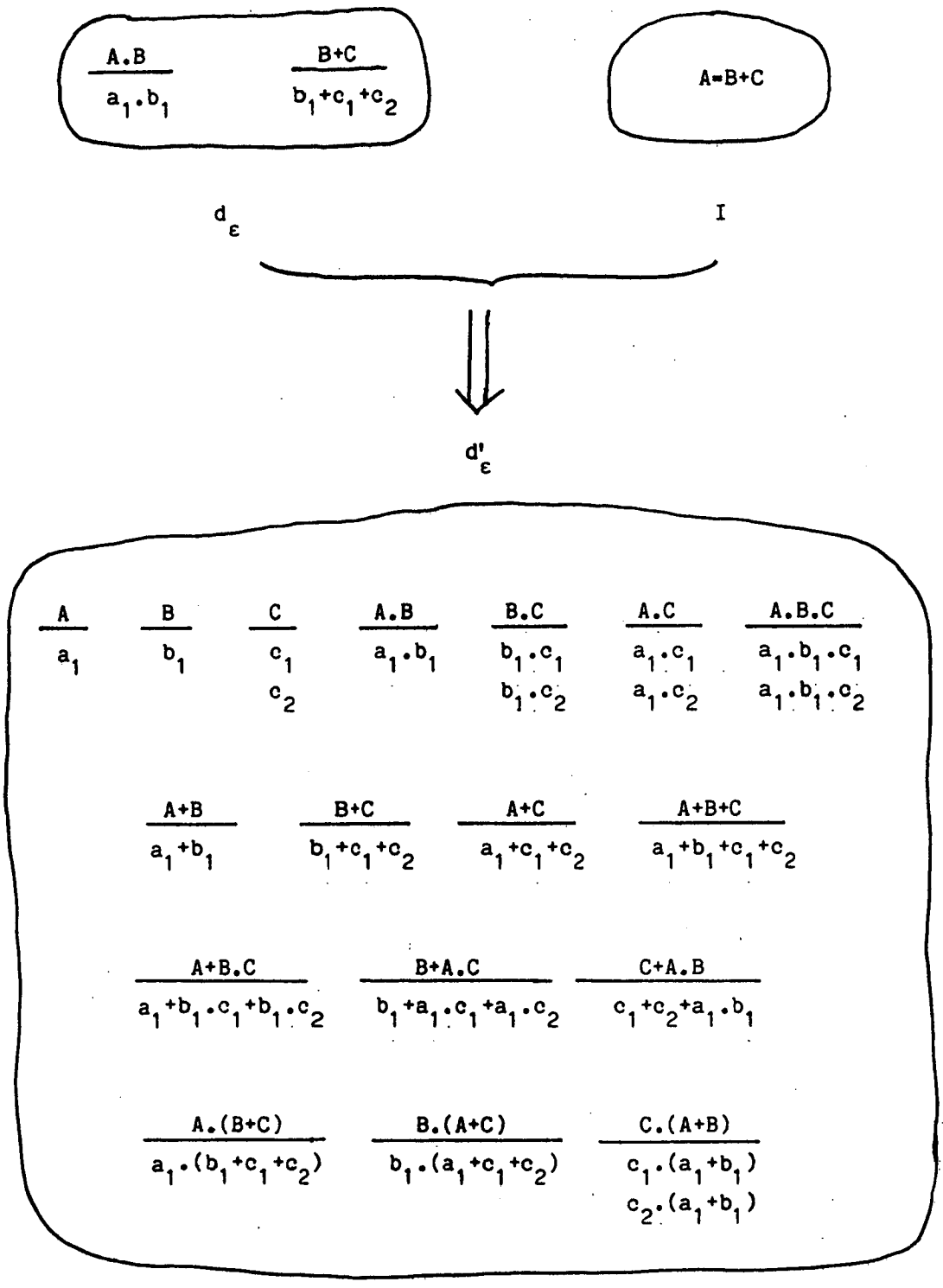


FIGURE 14 The given database d_ε and the external information I imply a new database d'_ε

Let us recall that the external information I consists of dependencies, assumptions, and model-oriented information. However, only the dependencies are stated explicitly. Thus, for example, when we write I = {A=B.C, B=C+D} this means that I consists of :

(1) The dependencies A=B.C and B=C+D

(2) Assumptions 1 and 2 if the database is conjunctive, or assumptions 1,2, and 3 if the database is disjunctive

(3) The definition of the model that is, whatever we have seen in Section 2

Let us also recall (from Section 2) that in order to define a fact it is enough to give its domain.

Example 1. Refer to Figure 14 and consider the following definitions of the atomic facts a_1, b_1, c_1 and c_2 :

$$\delta(a_1) = \{1,2,3\},\ \delta(b_1) = \{1,2\},\ \delta(c_1) = \{1\},\ \delta(c_2) = \{2,3\}$$

It is easy to check that :

(a) Assumption 1 is verified

(b) $b_1+c_1+c_2$ is a chain

(c) $a_1=b_1+c_1+c_2$, as required by the dependency A=B+C

It follows that d_ε is consistent with I □

Example 2. Let U = {A,B} be a universe. Consider a database d_ε and external information I defined as follows :

$$d_\varepsilon(A.B) = a.b + a.b' , \qquad I = \{A.B=A\}$$

We shall show that d_ε is not consistent with I. Indeed, suppose that a, b, and b' are atomic facts such that $d_\varepsilon \vdash I$. Then Assumption 1 implies that : $a.b \neq \phi$ and $a.b' \neq \phi$. On the other hand, as A.B=A, we have :

$(A.B=A \text{ and } a.b \neq \phi) \Rightarrow a \leq b$, thus

$(a \leq b \text{ and } b \neq b') \Rightarrow a.b' = \phi$, a contradiction

Therefore $d_\varepsilon \not\models I$. Note that we have used the unique name assumption (from Section 2) in our proof. Indeed, we have assumed that $b \neq b'$ □

Example 3. Let $U = \{A,B,C\}$ be a universe. Consider a database d_ε and external information I defined as follows :

$$d_\varepsilon(A.B) = a_1.b_1 + a_2.b_1 , \qquad d_\varepsilon(B+C) = (b_1+c_1) , \qquad I = \{A=B+C\}$$

We shall show that d_ε is not consistent with I. Indeed, suppose that a_1, a_2, b_1, and c_1, are atomic facts such that $d_\varepsilon \models I$. Assumption 1 implies that the facts a_1, a_2, b_1, c_1, $a_1.b_1$, and $a_2.b_1$, are all true. Assumption 3 implies that b_1+c_1 is a maximal chain. As $a_1.b_1$ is true, so is $a_1.(b_1+c_1)$. As $A=B+C$, it follows that $a_1=b_1+c_1$. Similarly, as $a_2.b_1$ is true, so is $a_2.(b_1+c_1)$. As $A=B+C$, it follows that $a_2=b_1+c_1$. Thus $a_1=a_2$, a contradiction to the unique name assumption (from Section 2). Therefore $d_\varepsilon \not\models I$ □

Example 4. Let $U = \{A,B\}$ be a universe. Consider a database d_ε and external information I defined as follows :

$$d_\varepsilon(A.B) = a_1.b_1+a_2.b_1, \qquad d_\varepsilon(A+B) = (a_1+b_1) + (a_2+b_2), \qquad I = \emptyset$$

We shall show that d_ε is not consistent with I. Indeed, suppose that a_1, a_2, b_1, and b_2, are atomic facts such that $d_\varepsilon \models I$. Assumption 1 implies that the fact $a_2.b_1$ is true. Assumption 3 implies that a_1+b_1 and a_2+b_2 are distinct maximal chains. It follows (from the definition of a maximal chain) that the fact $a_2.b_1$ is false, a contradiction. Therefore $d_\varepsilon \not\models I$ □

In the light of these examples it is evident that we need a procedure for proving facts true or false. Such a procedure would be used for consistency checking and deductive query answering.

3.4. QUERY ANSWERING

We have seen, in Section 2, that the set $D\varepsilon$ of all current databases is a bounded lattice. The first element of this lattice is the empty database e_ε and the last element is the full database f_ε. Given a database d_ε of $D\varepsilon$, consistent with an external information I, we have seen how we can derive a new database d'_ε of $D\varepsilon$ such that $d_\varepsilon \leq d'_\varepsilon$. In Figure 14 we see a typical example of such derivation. The database d'_ε consists of all facts that we can prove true based on d_ε and I. In a sense d'_ε is the "answer" to our quest for knowledge about the enterprise.

Definition 18. Let d_ε be a database of $D\varepsilon$ consistent with external information I. The answer on d_ε and I, denoted by $\alpha(d_\varepsilon,I)$, is the database of $D\varepsilon$ containing all facts that can be proved true based on d_ε and I □

For example, referring to Figure 14, we have : $\alpha(d_\varepsilon,I) = d'_\varepsilon$. Of course, it is conceivable that we may not be interested in the "whole" answer $\alpha(d_\varepsilon,I)$ but only in a specific partitioning.

Definition 19. Let U be a universe. Let d_ε be a database of $D\varepsilon$ consistent with external information I. Let d'_ε be the answer on d_ε and I, that is, $\alpha(d_\varepsilon,I) = d'_\varepsilon$. Let Q be a partitioning of L(U). The answer on Q, with respect to d_ε and I, denoted by $\alpha(Q,d_\varepsilon,I)$, is defined as follows :

$$\alpha(Q,d_\varepsilon,I) = d'_\varepsilon(Q) \qquad \square$$

We shall often refer to Q as a query and we shall write $\alpha(Q)$ instead of $\alpha(Q,d_\varepsilon,I)$, when no confusion is possible. Here are some examples of answers to queries from Figure 14 :

$$\alpha(A.C) = d'_\varepsilon(A.C) = a_1.c_1 + a_1.c_2$$
$$\alpha(A.B.C) = d'_\varepsilon(A.B.C) = a_1.b_1.c_1 + a_1.b_1.c_2$$
$$\alpha(A+B.C) = d'_\varepsilon(A+B.C) = (a_1+b_1.c_1+b_1.c_2)$$
$$\alpha(C.(A+B)) = d'_\varepsilon(C.(A+B)) = c_1.(a_1+b_1) + c_2.(a_1+b_1)$$

It is important to note that given a database d_ε the answer to a query Q depends on the available external information. If this information changes then so does (in general) the answer to query Q. Let us see an example. Consider the universe U = {A,B,C,D} and a database d_ε on U defined as follows :

$$d_\varepsilon(A.B.C) = a_1.b_1.c_1 + a_2.b_2.c_1$$
$$d_\varepsilon(A.B.D) = a_1.b_1.d_1 + a_3.b_2.d_1$$

This database is shown in Figure 15 along with three different external informations, denoted by I_1, I_2 and I_3. It is easy to check that d_ε is consistent with I_1, I_2 and I_3. Suppose now that we would like to compute the answer to query A.B.D. It follows from Assumption 2 that :

a_1, a_2, and a_3 are the only true facts of A_ε
b_1 and b_2 are the only true facts of B_ε
d_1 is the only true fact of D_ε

Therefore the only facts of $(A.B.D)_\varepsilon$ than can possibly be true are the following :

$$a_1.b_1.d_1,\ a_1.b_2.d_1,\ a_2.b_1.d_1,\ a_2.b_2.d_1,\ a_3.b_1.d_1,\ a_3.b_2.d_1$$

As the facts $a_1.b_1.d_1$ and $a_3.b_2.d_1$ are recorded in the database they are true (from Assumption 1). In order to decide which of the remaining facts are true and which are false, we use the available external information :

I_1 = {A.B=A, B.C=B} In this case the facts $a_1.b_2.d_1$, $a_2.b_1.d_1$, and $a_3.b_1.d_1$ can be proved false. Indeed, consider first the fact $a_1.b_2.d_1$. As $a_1.b_1.c_1$ is recorded in the database, the fact $a_1.b_1$ is true (from Assumption 1). Thus we obtain :

$$(A.B=A \text{ and } a_1.b_1 \neq \phi) \Rightarrow a_1.b_2 = \phi$$

Therefore $a_1.b_2.d_1$ is false. By similar arguments we can show that the facts $a_2.b_1.d_1$ and $a_3.b_1.d_1$ are false. However, the remaining fact $a_2.b_2.d_1$ cannot be shown true or false.

$I_2 = \{A.B=A,\ B.C=B,\ B.D=B\}$ In this case the external information consists of I_1 plus a new dependency, namely B.D=B. Therefore the facts $a_1.b_2.d_1$, $a_2.b_1.d_1$, and $a_3.b_1.d_1$ can be proved false as before. As for the remaining fact $a_2.b_2.d_1$ it can now be proved true. Indeed, as $a_2.b_2.c_1$ is recorded in the database, the fact $a_2.b_2$ is true (from Assumption 1). By a similar argument we find that the fact $b_2.d_1$ is true. Therefore we have :

$$(B.D=B \text{ and } b_2.d_1 \neq \phi) \Rightarrow b_2.d_1 = b_2$$

It follows that $a_2.b_2.d_1 = a_2.b_2$. As the fact $a_2.b_2$ is true, we obtain that the fact $a_2.b_2.d_1$ is true too.

$I_3 = \{A.B=A,\ B.C=B,\ A.B.D=B.D\}$ In this case the external information consists of I_1 plus a new dependency, namely B.D.A = B.D. Therefore the facts $a_1.b_2.d_1$, $a_2.b_1.d_1$, and $a_3.b_1.d_1$ can be proved false as before. As for the remaining fact $a_2.b_2.d_1$ it can now be proved false. Indeed, it follows from Assumption 1 that $a_3.b_2.d_1$ is true, so we have :

$$(A.B.D = B.D \text{ and } a_3.b_2.d_1 \neq \phi) \Rightarrow b_2.d_1 \leq a_3$$

As $a_2 \neq a_3$ it follows that the fact $a_2.b_2.d_1$ is false.

The example of Figure 15 shows clearly that the answer to a query may not be "complete". Indeed, in the first case (with external information I_1) we cannot prove the fact $a_2.b_2.d_1$ true, and this is why we do not include it in the answer to query A.B.C. However, we cannot prove $a_2.b_2.d_1$ false either ! In other words, the external information I_1 is not sufficient in order to decide which facts of A.B.D are true and which are false. This problem does not arise in the remaining two cases (external information I_2 and I_3), and so the answers to query A.B.C are "complete" in those cases.

A very important notion related to query answering is that of equivalence. We have seen that every pair (d_ε, I), such that $d_\varepsilon \models I$, is associated with an answer, denoted by $\alpha(d_\varepsilon, I)$. This answer is a

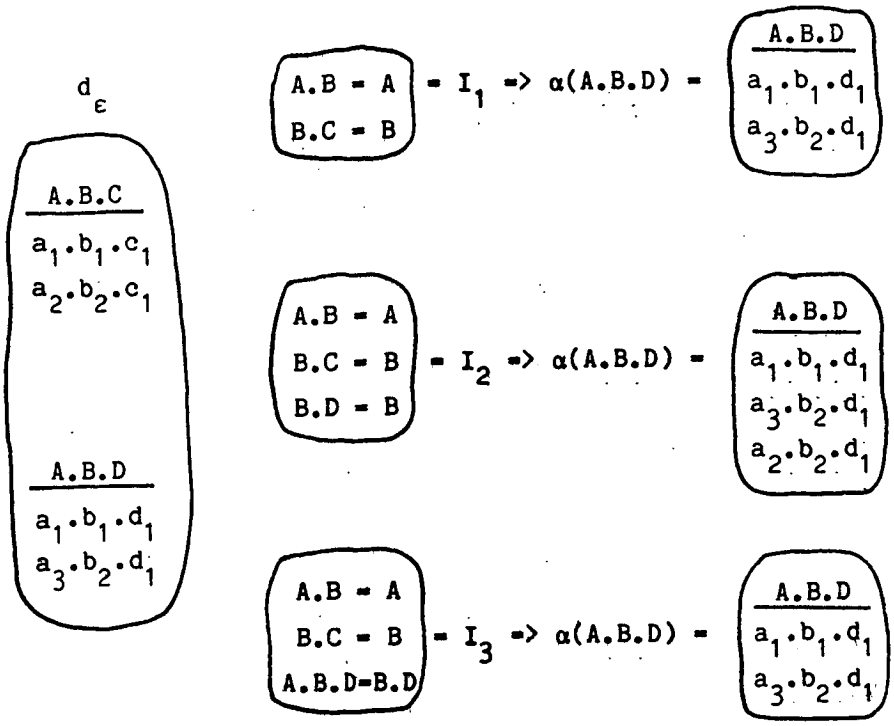


FIGURE 15. Query answering on a database d_ε under varying external information.

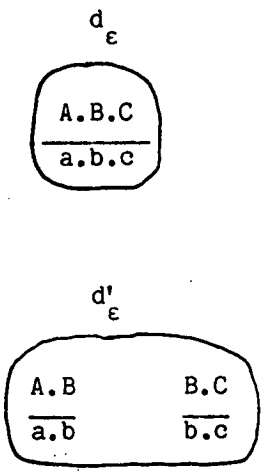


FIGURE 16. $(d_\varepsilon, I) \equiv (d'_\varepsilon, I')$ whereas $(d'_\varepsilon, I) \leq (d_\varepsilon, I)$ and $(d'_\varepsilon, I) \not\equiv (d_\varepsilon, I)$

database of Dε representing the information content of (d_ε, I). We shall say that two pairs, (d_ε, I) and (d'_ε, I'), are equivalent if they have the same information content. In order to simplify matters we assume that : if $d_\varepsilon \not\models I$ then $\alpha(d_\varepsilon, I) = e_\varepsilon$, where e_ε is the empty database.

Definition 20. Let d_ε and d'_ε be databases of Dε. Let I and I' be external informations. We say that (d_ε, I) is <u>equivalent</u> to (d'_ε, I'), denoted by $(d_\varepsilon, I) \equiv (d'_\varepsilon, I')$, if $\alpha(d_\varepsilon, I) = \alpha(d'_\varepsilon, I')$ □

This definition can also be given (equivalently) in terms of answers to individual queries as follows :

$$(d_\varepsilon, I) \equiv (d'_\varepsilon, I') \iff \forall\, Q \in L(U) \quad \alpha(Q, d_\varepsilon, I) = \alpha(Q, d'_\varepsilon, I')$$

For example, let $U = \{A,B,C\}$ and consider two databases d_ε and d'_ε defined as follows :

$$d_\varepsilon(A.B.C) = a.b.c$$
$$d'_\varepsilon(A.B) = a.b \ , \quad d'_\varepsilon(B.C) = b.c$$

These databases are shown in Figure 16. Let I be the external information on d_ε and suppose that I contains no dependencies other than the trivial ones (this is denoted by writing I=∅). Let I' be the external information on d'_ε and suppose that $I' = \{B=B.C\}$. It is easy to check that $(d_\varepsilon, I) \equiv (d'_\varepsilon, I')$ by computing the answers to all individual queries. Here are some examples :

$$\alpha(A.B, d_\varepsilon, I) = a.b = \alpha(A.B, d'_\varepsilon, I')$$
$$\alpha(A.B.C, d_\varepsilon, I) = a.b.c = \alpha(A.B.C, d'_\varepsilon, I')$$
$$\alpha(A+C, d_\varepsilon, I) = a+c = \alpha(A+C, d'_\varepsilon, I')$$

On the other hand, the pair (d_ε, I) is not equivalent to the pair (d'_ε, I) as

$$\alpha(A.B.C, d_\varepsilon, I) = a.b.c \neq \alpha(A.B.C, d'_\varepsilon, I) = \phi$$

Let $\mathcal{J}$ denote the set of all possible external informations. Let α be a function from the product set $D\varepsilon \times \mathcal{J}$ into the set $D\varepsilon$ that associates each pair (d_ε, I) in $D\varepsilon \times \mathcal{J}$ with its answer $\alpha(d_\varepsilon, I)$ in $D\varepsilon$. It follows from Definition 20 that two pairs (d_ε, I) and (d'_ε, I') are equivalent if they have the same image under α. Let $[(d_\varepsilon, I)]$ denote the equivalence class of (d_ε, I) that is,

$$[(d_\varepsilon, I)] = \{(d'_\varepsilon, I') \mid \alpha(d_\varepsilon, I) = \alpha(d'_\varepsilon, I')\}$$

Let $(D\varepsilon \times \mathcal{J})/\alpha$ denote the set of all equivalence classes of $D\varepsilon \times \mathcal{J}$. It is easy to see that

$$(D\varepsilon \times \mathcal{J})/\alpha = \{\alpha^{-1}(d_\varepsilon) \mid d_\varepsilon \in D\varepsilon\}$$

The set $(D\varepsilon \times \mathcal{J})/\alpha$ can be ordered using the lattice ordering of $D\varepsilon$.

Definition 21. Let d_ε and d'_ε be databases of $D\varepsilon$. Let I and I' be external informations. We say that the class $[(d_\varepsilon, I)]$ is <u>less defined</u> than the class $[(d'_\varepsilon, I')]$, denoted by $[(d_\varepsilon, I)] \leq [(d'_\varepsilon, I')]$, if $\alpha(d_\varepsilon, I) \leq \alpha(d'_\varepsilon, I')$ □

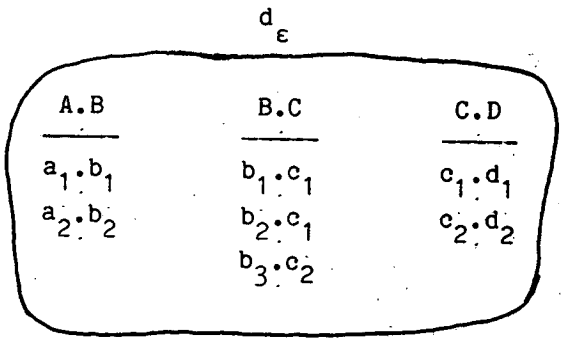
For notational convenience we shall often write $(d_\varepsilon, I) \leq (d'_\varepsilon, I')$ instead of $[(d_\varepsilon, I)] \leq [(d'_\varepsilon, I')]$. Definition 21 can also be given (equivalently) in terms of answers to individual queries as follows :

$$(d_\varepsilon, I) \leq (d'_\varepsilon, I') \iff \forall\, Q \in L(U) \quad \alpha(Q, d_\varepsilon, I) \leq \alpha(Q, d'_\varepsilon, I')$$

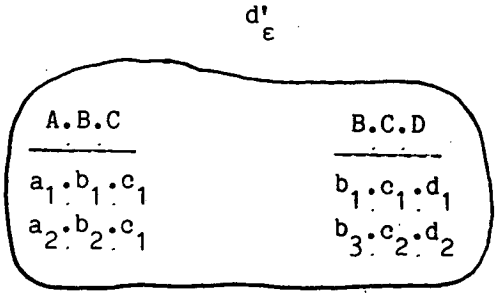
In the example of Figure 16 it is easy to check that $(d'_\varepsilon, I) \leq (d_\varepsilon, I)$. An immediate consequence of definitions 20 and 21 is the following :

$$(d_\varepsilon, I) \equiv (d'_\varepsilon, I') \iff (d_\varepsilon, I) \leq (d'_\varepsilon, I') \text{ } \underline{\text{and}} \text{ } (d'_\varepsilon, I') \leq (d_\varepsilon, I)$$

The examples of figures 17 and 18 should help clarify further the notions of equivalence and ordering that we have introduced in this section.

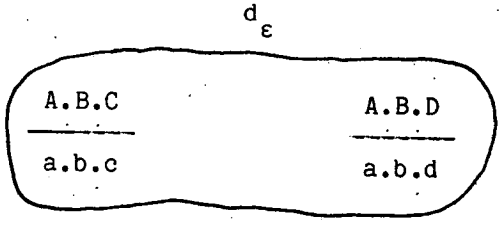


$I = \{B=B.A\}$

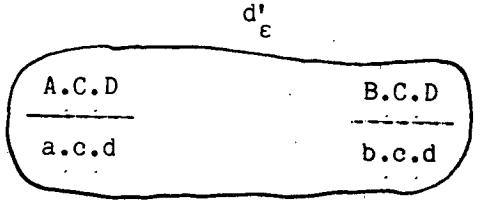


$I' = \emptyset$

FIGURE 17. $(d_\varepsilon, I) \leq (d'_\varepsilon, I')$



$I_1 = \emptyset$
$I_2 = \{B=B.C\}$



$I'_1 = \emptyset$
$I'_2 = \{C=C.B\}$

FIGURE 18. (d_ε, I_1) and (d'_ε, I'_1) are not comparable whereas :
$(d_\varepsilon, I_1) \leq (d'_\varepsilon, I'_2)$, $(d'_\varepsilon, I'_1) \leq (d_\varepsilon, I_2)$, and $(d_\varepsilon, I_2) \equiv (d'_\varepsilon, I'_2)$

The problem of comparing two pairs (d_ε, I) and (d'_ε, I'), with respect to their information content, is of central interest in database theory. For instance, when we know that two or more pairs are equivalent then we are faced with a choice among alternative representations of the same information. Some choices are more convenient than others for various reasons. In the following section we study some aspects of the equivalence problem for an important class of databases namely, conjunctive databases.

4. CONJUNCTIVE DATABASES

In this section we discuss some aspects of equivalence in conjunctive databases. In our discussions we assume a universe U and we denote by CL(U) the closure of U under product. For example, if U = {A,B,C} then we have :

CL(U) = {A, B, C, A.B, B.C, A.C, A.B.C}

The set CL(U) consists of all the conjunctive partitionings of L(U). Therefore the domain of every conjunctive database on U is a subset of CL(U). We denote by CDε the set of all conjunctive databases on ε and U. We restrict our attention to conjunctive dependencies that is, dependencies of the form q=r such that q and r are instances of partitionings of CL(U). We consider only conjunctive queries that is, queries from CL(U). In summary, our assumptions in this section are the following :

- the database d_ε is conjunctive ,
- the external information I consists of conjunctive dependencies, assumptions 1 and 2, and model-oriented information, and
- the queries are conjunctive

Let us recall that model-oriented information is whatever we have seen in Section 2. For example, for all Q in CL(U), the equation Q=Q.Q is model-oriented information, as it is a consequence of the definition

of the model (see Proposition 1, Section 2). Thus for all Q in CL(U) the trivial conjunctive dependency Q=Q.Q is in I. However, trivial dependencies are not stated explicitly in our examples.

In Section 4.1 we define expansion and reduction of a partitioning with respect to a given external information I. Subsequenctly, we use these concepts, in Section 4.2, in order to determine equivalence of conjunctive databases.

4.1. EXPANSION AND REDUCTION

Throughout this section we assume that all dependencies contained in the external information I are conjunctive. We denote by I^+ the external information I augmented by all conjunctive dependencies that are consequencies of the dependencies in I. Let us use the following as a "running" example :

$U = \{A,B,C,D,E\}$, $I = \{A.B=A.B.C\ ,\ C=C.A\ ,\ D.E = D.E.B.C\ ,\ B.E=B.E.D\}$

The dependency B.C=B.C.A is a consequence of the dependency C=C.A and thus it is in I^+. As a second example, consider the dependency D.E=D.E.B. This dependency is a consequence of the dependency D.E = D.E.B.C because :

$$D.E = D.E.B.C \Rightarrow D.E \leq D.E.B.C \Rightarrow D.E \leq D.E.B \Rightarrow D.E = D.E.B$$

Thus the dependency D.E = D.E.B is in I^+. Now, let us recall that the information carried by an equation such as B.C = B.C.A is the following : the partitioning (denoted by) B.C and the partitioning (denoted by) B.C.A are the same partitioning. Therefore we can view B.C as a "reduced" notation and B.C.A as an "expanded" notation for the same partitioning. Carrying this idea one step further, consider the partitioning A.B and suppose B = B.C. Then we can say that A.B "expands" to B.C in the sense that the product of A.B and B.C provides an "expanded" notation for A.B. Indeed, if B = B.C then A.B = A.B.C. In the opposite direction, we can say that A.B.C "reduces" to A.B, in the sense that A.B is a "reduced" notation for A.B.C.

Definition 22. Let Q and R be two partitionings of CL(U). Let I be a given external information. We say that Q expands to R, or that R reduces to Q, with respect to I, if there are partitionings Q_1, X, R_1 in CL(U) such that : $Q=Q_1.X$, $R=X.R_1$, and $X=X.R_1$ are in I^+. □

Clearly, if Q=R then Q expands to Q.R (trivially). In our running example, A.B expands to A.B.C, C expands to C.A, D.E expands to D.E.B.C, and B.E expands to B.E.D. On the other hand, A.B.E expands to A.B.C (by A.B = A.B.C) and to B.E.D (by B.E = B.E.D). Similarly, A.D.E expands to D.E.B.C (by D.E = D.E.B.C), and D.E.B.C expands to C.A (by C = C.A).

Suppose that Q expands to R and let $Q.R = (Q_1.X).(X.R_1)$, as in Definition 22. Let $q_1.x.r_1$ be a true fact of Q.R. It follows that : $q_1.x$ is a true fact of $Q_1.X$ and $x.r_1$ is a true fact of $X.R_1$. Conversely, suppose that $q_1.x$ is a true fact of $Q_1.X$ and that $x.r_1$ is a true fact of $X.R_1$. As $X=X.R_1$, it follows that $q_1.x.r_1 = q_1.x$ and, therefore, $q_1.x.r_1$ is a true fact of Q.R ; moreover, $x.r_1'$ is a false fact of $X.R_1$, for every true fact r_1' of R_1 ($r_1' \neq r_1$). It follows that $q_1.x.r_1'$ is false for every $r_1' \neq r_1$. We state this result formally in the following :

Lemma 1. Suppose that Q expands to R and let $Q.R = (Q_1.X).(X.R_1)$, as in Definition 22. If $q_1.x$ is a true fact of Q and $x.r_1$ is a true fact of R then

(i) $q_1.x.r_1$ is a true fact of Q.R
(ii) $q_1.x.r_1'$ is a false fact of Q.R, for all $r_1' \neq r_1$ □

Given a partitioning S of CL(U), a relevant question is whether there is a "shortest" reduction and a "longest" expansion of S. In our running example, we have seen that A.B.C reduces to A.B. On the other hand, A.B is irreducible, with respect to I, as it does not reduce any further (neither A=A.B nor B = B.A is in I^+). However, A.B.C reduces also to B.C, and B.C is again irreducible. It follows that, given a partitioning S, there may be more than one "shortest" reduction of S. If Q is irreducible and Q expands to R then Q is called a key of Q.R.

In our running example, A.B and B.C are keys of A.B.C. Similarly, B.E is a key of B.D.E, and A.B.E is a key of A.B.C.D.E. On the other hand, D.E is irreducible but it is not a key of A.B.C.D.E, as D.E does not expand to A.B.C.D.E. However, D.E expands to B.C.D.E and thus D.E is a key of B.C.D.E. Let us note that all atoms of U are irreducible. Thus, in our running example, C is a key of A.C, as C is irreducible and it expands to A.C.

We have just seen that, given a partitioning S, there may be more than one "shortest" reduction of S. However, there is always a unique "longest" expansion. First, let us note that if Q expands to R then Q expands also to Q.R (trivially). On the other hand, if Q expands to R and to R' then Q expands to R.R'. Indeed, following Definition 22, let us suppose that : $Q = Q_1.X = Q'_1.X'$, $R=X.R_1$, $R'=X'.R'_1$, $X=X.R_1$, and $X'=X'.R'_1$. Then we have :

$$Q = (Q_1,Q'_1).(X.X'), \quad R.R' = (X.X').(R_1.R'_1), \quad X.X' = (X.X').(R_1.R'_1)$$

It follows that Q expands to R.R'. As a consequence, there can be only one "longest" expansion of Q. In our running example, A.B expands to A.B.C. On the other hand, A.B.C is <u>saturated</u> as it does not expand any further (the only possibility is C=C.A but A appears already in A.B.C). It follows from our discussion so far that every partitioning Q of CL(U) has a unique saturated expansion. We call this unique saturated expansion of Q the <u>closure</u> of Q and we denote it by Q^+. Here are some examples of closures :

$$(A.B)^+ = A.B.C \ , \ (A.C)^+ = A.C \ , \ (A.B.E)^+ = A.B.C.D.E \ , \ (A.E)^+ = A.E$$

Note that A.E is both, irreducible and saturated. A partitioning that is both irreducible and saturated is called a <u>prime</u>. Note also that the atoms of U are always irreducible and that glb(U) is always saturated, for any external information I. Clearly, if I contains only trivial dependencies then every partitioning of CL(U) is both, irreducible and saturated that is, no reduction or expansion is possible.

If a partitioning is not saturated then it can expand, possibly in many different ways. Let us consider the following example :

$U = \{A,B,C,D,E,F,G\}$, $I = \{B=B.A,\ CD = C.D.E,\ C=C.F,\ E=E.G\}$

The partitioning B.C.D expands to B.A but B.A does not expand any further (it is saturated). However, B.C.D also expands to C.D.E, and C.D.E is not saturated. C.D.E expands, in turn, to C.F and to E.F.G which are both saturated. Thus we have :

$$\begin{aligned} B.C.D &= (B.C.D).(B.A) \\ &= (B.C.D).(C.D.E).(C.F) \\ &= (B.C.D).(C.D.E).(E.F.G) \end{aligned}$$

Each of these products represents a different way of expanding B.C.D up to a saturated partitioning.

Definition 23. A sequence $< S_1,S_2,\ldots,S_n >$ of partitionings in CL(U) is an expanding sequence if S_i expands to S_{i+1}, for all $i=1,2,\ldots,n-1$ (assume $n > 1$) □

In our previous example the following are expanding sequences :

< B.C.D, B.A > , < B.C.D, C.D.E, C.F > , < B.C.D, C.D.E, E.F.G >

An expanding sequence is maximal if its first term is irreducible and its last term is saturated. In our previous example all three expanding sequences are maximal. Let us note that if $< S_1,S_2,\ldots,S_n >$ is an expanding sequence then any subsequence of the form $< S_i,S_{i+1},\ldots,S_j >$ is also an expanding sequence. Let us also note that $S_1.S_2\ldots S_i$ expands to S_{i+1}, for all $i=1,2,\ldots,n-1$. Using this fact and a simple induction argument we can extend the result of Lemma 1 as follows :

Lemma 2. Let $<S_1,S_2,\ldots,S_n >$ be an expanding sequence. Let $S_i.S_{i+1} = (Q_i.X_i).(X_i.Q_{i+1})$, where $X_i = X_i.Q_{i+1}$ for all $i=1,2,\ldots,n-1$

(see also Definition 22). If $s_i = q_i \cdot x_i$ is a true fact of S_i and $s_{i+1} = x_i \cdot q_{i+1}$ is a true fact of S_{i+1}, i=1,2,...,n-1, then we have :

(i) $s_1 \cdot s_2 \cdots s_n$ is a true fact of $S_1 \cdot S_2 \cdots S_n$

(ii) $s_1 \cdot s_2 \cdots s_{i-1} \cdot s'_i \cdot s_{i+1} \cdots s_n$ is a false fact, for all $s'_i \neq s_i$, i=2,...,n □

Given a partitioning Q of CL(U), we call Q-sequence any expanding sequence whose first term is Q; we call Q-expansion the set of all terms in one or more Q-sequences. For example, consider the following expanding sequences (see Figure 19) :

< B.C.D, A.B > , < B.C.D, C.D.E, C.F > , < B.C.D, C.D.E, E.F.G >

These sequences are B.C.D-sequences. The set of all terms appearing in these sequences is the following :

{B.C.D, A.B, C.D.E, C.F, C.D.E, E.F.G}

This set is a B.C.D-expansion. The B.C.D-sequences contained in this set can be conveniently represented using a graph in which an arrow from Q to R means that Q expands to R. The B.C.D-expansion that we have just seen is represented by graph (a) in Figure 19. Using Lemma 2 and a simple induction argument, on the number of Q-sequences in a Q-expansion, we obtain the following result :

Proposition 5. Let Q be a partitioning of CL(U). Let $\{Q=Q_1, Q_2, \ldots, Q_m\}$ be a Q-expansion. If $s=s_1 \cdot s_2 \cdots s_m$ is a true fact of $Q_1 \cdot Q_2 \cdots Q_m$ then $s_1 \cdot s_2 \cdots s_{i-1} \cdot s'_i \cdot s_{i+1} \cdots s_m$ is a false fact, for all $s'_i \neq s_i$, i=2,...,m □

4.2. EQUIVALENCE

Let d_ε and d'_ε be conjunctive databases. Let I and I' be external informations. In the light of our discussion of equivalence in Section 3.4, we can specialize the definitions that we have seen there to conjunctive databases. Thus we can write :

$U = \{A,B,C,D,E,F,G\}$, $I = \{B=B.A,\ C.D=C.D.E,\ C=C.F,\ E=E.F.G\}$

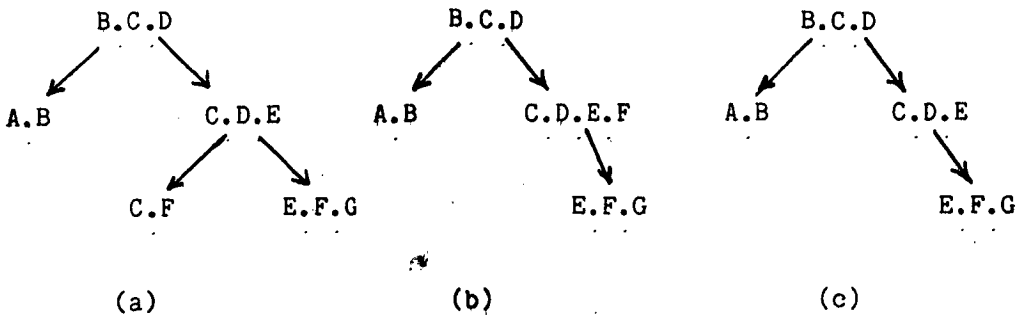
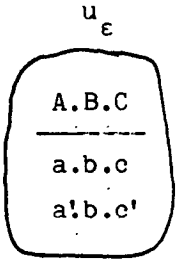
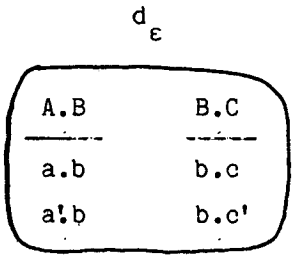


FIGURE 19. Three B.C.D-expansions represented by graphs.



$U = \{A,B,C\}$

$I = \{A=A.C\}$



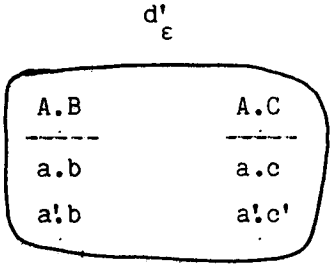


FIGURE 20. $(d_\varepsilon, I) \leq (u_\varepsilon I)$

$$(d_\varepsilon,I) \equiv (d'_\varepsilon,I') \iff \forall\ Q\in CL(U)\quad \alpha(Q,d_\varepsilon,I) = \alpha(Q,d'_\varepsilon,I')$$

$$(d_\varepsilon,I) \leq (d'_\varepsilon,I') \iff \forall\ Q\in CL(U)\quad \alpha(Q,d_\varepsilon,I) \leq \alpha(Q,d'_\varepsilon,I')$$

In this section we study the problem of deciding whether two pairs, (u_ε,I) and (d_ε,I), are equivalent under the following assumptions :

A1 $\quad dom(u_\varepsilon) = \{glb(U)\}$

A2 $\quad glb(dom(d_\varepsilon)) = glb(U)$

A3 $\quad \forall\ Q\in dom(d_\varepsilon)\quad d_\varepsilon(Q) = \alpha(Q,u_\varepsilon,I)$

The first assumption says that the domain of u_ε contains a single partitioning namely, the product of all atoms in U. Such a database is called a universal database. The second assumption says that the domain of d_ε can contain more than one partitioning, provided that the greatest lower bound of all partitionings in the domain is equal to glb(U). The third assumption says that, for all Q in $dom(d_\varepsilon)$, the instance of Q associated with Q, under d_ε, is the answer to Q obtained from (u_ε,I). A database d_ε satisfying the assumptions A2 and A3 is called a decomposition of u_ε. Figure 20 shows a universal database u_ε and a decomposition d_ε of u_ε. We can now restate the problem that we study in this section as follows : decide whether $(u_\varepsilon,I) \equiv (d_\varepsilon,I)$, where u_ε is a universal database and d_ε is a decomposition of u_ε. It is important to note that the external information I is common to both databases. Let us also note that (d_ε,I) is less defined than (u_ε,I). This follows immediately from assumption A3.

In the specific problem that we consider in this section, the definition of equivalence can be written as follows :

$$(u_\varepsilon,I) \equiv (d_\varepsilon,I) \iff \alpha(glb(U),u_\varepsilon,I) = \alpha(glb(U),d_\varepsilon,I)$$

This is an immediate consequence of assumptions A1, A2, and A3. For example, in Figure 20, where glb(U) = A.B.C we find :

$$\alpha(A.B.C,u_\varepsilon,I) = a.b.c + a'.b.c' \neq \alpha(A.B.C,d_\varepsilon,I) = \phi$$

It follows that (u_ε,I) is not equivalent to (d_ε,I). Now, let us define (in Figure 20) a new decomposition d'_ε of u_ε as follows :

$$d'_\varepsilon(A.B) = \alpha(A.B,u_\varepsilon,I) = a.b + a'.b$$
$$d'_\varepsilon(A.C) = \alpha(A.C,u_\varepsilon,I) = a.c + a'.c'$$

Let us note that A.B expands to A.C with respect to I. Therefore we can apply Lemma 1 (from Section 4.1) to conclude that $\alpha(A.B.C,u_\varepsilon,I) = \alpha(A.B.C,d'_\varepsilon,I)$. It follows that $(u_\varepsilon,I) \equiv (d'_\varepsilon,I)$. Thus Lemma 1, or its more general Proposition 5, provides the following sufficient condition for equivalence : $(u_\varepsilon,I) \equiv (d_\varepsilon,I)$, if there is Q in $dom(d_\varepsilon)$ such that $dom(d_\varepsilon)$ is a Q-expansion with respect to I. We state this result formally in the following proposition :

Proposition 6. Let u_ε be a universal database, and let d_ε be a decomposition of u_ε. Let I be a given external information. Then $(u_\varepsilon,I) \equiv (d_\varepsilon,I)$ if there is Q in $dom(d_\varepsilon)$ such that $dom(d_\varepsilon)$ is a Q-expansion, with respect to I. □

When (d_ε,I) is equivalent to (u_ε,I) then we say that (d_ε,I) is a lossless decomposition of (u_ε,I). Proposition 6 gives a sufficient condition for lossless decomposition. It is important to note that this condition depends on the external information I and on the domain of d_ε but it is independent of the current instance of u_ε.

Let us refer back to Figure 19 for some examples of lossless decompositions. First, let us recall that the graphs (a), (b), and (c), that we see in that figure, represent three different B.C.D-expansions. Now, let us consider a universal database u_ε and three decompositions d_ε, d'_ε, and d''_ε, such that :

$$dom(u_\varepsilon) = \{A.B.C.D.E.F.G\}$$
$$dom(d_\varepsilon) = \{B.C.D,\ A.B,\ C.D.E,\ C.F,\ E.F.G\}$$
$$dom(d'_\varepsilon) = \{B.C.D,\ A.B,\ C.D.E.F,\ E.F.G\}$$
$$dom(d''_\varepsilon) = \{B.C.D,\ A.B,\ C.D.E,\ E.F.G\}$$

The domains of d_ε, d'_ε, and d''_ε correspond to graphs (a), (b), and (c), respectively. Each domain contains the partitioning B.C.D and it is a B.C.D-expansion. It follows from Proposition 6 that all three decompositions are lossless. Let us note that $dom(d''_\varepsilon) \subsetneq dom(d_\varepsilon)$. Let us also note that the three decompositions d_ε, d'_ε, and d''_ε correspond to three different "factorizations" of the domain of u_ε. Indeed,

$$
\begin{aligned}
A.B.C.D.E.F.G &= (B.C.D).(A.B).(C.D.E).(C.F).(E.F.G) && \text{from graph (a)}\\
&= (B.C.D).(A.B).(C.D.E.F).(E.F.G) && \text{from graph (b)}\\
&= (B.C.D).(A.B).(C.D.E).(E.F.G) && \text{from graph (c)}
\end{aligned}
$$

Let us recall that if the external information I contains only trivial dependencies then no expansion is possible. It follows that no non trivial lossless decomposition is possible. On the other hand, if I contains nontrivial dependencies then more than one lossless decomposition may be possible (see Figure 19). In this case we are faced with a choice among alternative representations of the same information. Some choices are more convenient than others for various reasons. We discuss here briefly three examples of desirable properties for lossless decompositions. In our discussions, u_ε denotes a universal database and $d_\varepsilon, d'_\varepsilon, \ldots,$ denote decompositions of u_ε.

Minimality. The partitionings in the domain of a lossless decomposition d_ε are often required to be of a specific form. One such requirement is miniality. Let Q and R be two partitionings of CL(U) and suppose that Q expands to R. Then there are partitionings Q_1, X, and R_1, such that : $Q = Q_1.X$, $R = X.R_1$, and $X = X.R_1$ (see Definition 22). R is called a minimal expansion of Q if R_1 is irreducible. An expanding sequence $< Q_1, Q_2, \ldots, Q_n >$ is minimal if

(a) Q_1 is a minimal expansion of an irreducible partitioning, and

(b) Q_i is a minimal expansion of Q_{i-1}, $i=2,\ldots,n$

A Q-expansion is minimal if all its Q-sequences are minimal. For example, referring back to Figure 19, it is easy to see that the

following B.C.D-sequences are all minimal :

< B.C.D, A.B >, < B.C.D, C.D.E, C.F >, < B.C.D, C.D.E, E.F.G >

Therefore, the B.C.D-expansion made up from all terms of these sequences is minimal. This B.C.D-expansion is represented by graph (a) in Figure 19. The B.C.D-expansion represented by graph (b) is not minimal, as C.D.E.F is not a minimal expansion of B.C.D (the dependency E=E.F is in I^+ and, therefore, E.F is reducible). Finally, the B.C.D-expansion represented by graph (c) is minimal as all its B.C.D-sequences are minimal. A lossless decomposition d_ε of u_ε is minimal if there is Q in $dom(d_\varepsilon)$ such that $dom(d_\varepsilon)$ is a minimal Q-expansion. Thus in Figure 19, graphs (a), (b), and (c), correspond to lossless decompositions, of which (a) and (c) are also minimal. Clearly, if there is a (nontrivial) lossless decomposition of u_ε then there is a minimal lossless decomposition of u_ε. In order to find a minimal lossless decomposition of u_ε, we can

- either start with a minimal expansion of a key of glb(U) and expand it "minimally"
- or, start with glb(U) and remove atoms "minimally"

The resulting algorithms and their analyses are presented in a separate paper [5].

Nonredundancy. A decomposition d_ε of u_ε is called <u>redundant</u> if there is decomposition d'_ε of u_ε such that $dom(d'_\varepsilon) \subsetneq dom(d_\varepsilon)$. A desirable property of a lossless decomposition d_ε of u_ε is that it be nonredundant. For example, referring back to Figure 19, we see three lossless decompositions corresponding to graphs (a), (b), and (c). Let us denote these decompositions by d_ε, d'_ε, d''_ε, respectively. Thus we have :

$dom(d_\varepsilon)$ = {B.C.D, A.B, C.D.E, C.F, E.F.G}
$dom(d'_\varepsilon)$ = {B.C.D, A.B, C.D.E.F, E.F.G}
$dom(d''_\varepsilon)$ = {B.C.D, A.B, C.D.E, E.F.G}

As we have seen earlier, all three decompositions are lossless. However, as $dom(d''_\varepsilon)$ is a proper subset of $dom(d_\varepsilon)$, the decomposition d_ε is redundant. On the other hand, the decomposition d'_ε is nonredundant, because if we remove any partitioning from $dom(d'_\varepsilon)$ then what remains is no more a decomposition of u_ε. For the same reason d''_ε is nonredundant too. Let us recall, from our earlier discussion, that d''_ε is also minimal. Thus d''_ε is a lossless, minimal, and nonredundant decomposition of u_ε. Clearly, if there is a (nontrivial) lossless decomposition of u_ε then there is a nonredundant lossless decomposition of u_ε. In order to find a nonredundant lossless decomposition of u_ε we can proceed as follows :

(1) start with a Q-expansion of a lossless decomposition d_ε

(2) remove a partitioning R if

  (a) $glb(dom(d_\varepsilon)-\{R\}) = glb(U)$ and

  (b) there is S in $(dom(d_\varepsilon)-\{R\})$ such that $(dom(d_\varepsilon)-\{R\})$ is an S-expansion

(3) repeat (2) until no partitioning can be removed.

For example, referring back to Figure 19, let d_ε, d'_ε, and d''_ε denote the lossless decompositions corresponding to graphs (a), (b), and (c). We have seen earlier that d_ε is redundant. Applying the algorithm just described, we find that C.F is the only partitioning than can be removed. The resulting nonredundant lossless decomposition is d'_ε.

Dependency preservation. Let d_ε be a decomposition of u_ε. Let I be a given external information. It follows from the definition of a decomposition that : if $u_\varepsilon \models I$ then $d_\varepsilon \models I$. However, the converse is not necessarily true, as the example of Figure 21 shows. Indeed, the decomposition d_ε of Figure 21 satisfies I, if the atomic facts a,b,c and c', are defined as follows :

$$\delta(a) = \{1,2\},\ \delta(b) = \{2,3,4\},\ \delta(c) = \{3,5\},\ \delta(c') = \{4,6\}$$

On the other hand, there is no definition of the atomic facts a,b,c, and c', such that u_ε satisfies I. Indeed, suppose that a,b,c, and c' are atomic facts such that $u_\varepsilon \models I$. Then Assumption 1 (from Section 3) implies that : $a.c \neq \phi$ and $a.c' \neq \phi$. On the other hand, as A=A.C, we have :

$$(A=A.C \text{ and } a.c \neq \phi) \Rightarrow a \leq c, \text{ thus}$$
$$(a \leq c \text{ and } c \neq c') \Rightarrow a.c' = \phi, \text{ a contradiction}$$

Therefore $u_\varepsilon \not\models I$. Thus d_ε satisfies I whereas u_ε does not. A desirable property of a decomposition d_ε of u_ε is the following : if d_ε satisfies I then u_ε satisfies I. Such a decomposition of u_ε is called dependency preserving, with respect to I. Let us note that if each dependency of I is "embedded" in a partitioning of $dom(d_\varepsilon)$ then d_ε is dependency preserving. We say that a dependency Q=R is embedded in a partitioning S if $S \leq Q.R$. For example, the dependency B.C=C.D is embedded in the partitioning A.B.C.D. More generally, if each dependency in a "cover" of I is embedded in a partitioning of $dom(d_\varepsilon)$ then d_ε is dependency preserving. A subset J of I^+ is called a cover of I if all dependencies in I are consequences of those in J. For example, let

$$I = \{A=A.B.C,\ C=C.A,\ A=A.D,\ D=D.E,\ A=A.E\}$$
$$J = \{A=A.B.C,\ C=C.A,\ C=C.D,\ D=D.E\}$$

The set J is a subset of I^+, as its dependency C=C.D is a consequence of the dependencies C=C.A and A=A.D of I. On the other hand, J is a cover of I as the dependencies of I are consequences of the dependencies of J. In the example of Figure 21 the dependency A=A.C is embedded in the partitioning A.C. It follows that the decomposition d'_ε, of Figure 21, is dependency preserving. Note that d'_ε is also lossless (A.B expands to A.C). Unfortunately, it is not always possible to find a lossless decomposition of u_ε which is also dependency preserving. Consider the following example :

$$U = \{A,B,C\}\ ,\ I = \{A.B=A.B.C,\ C=C.B\}$$

The dependency A.B=A.B.C can be embedded only in A.B.C. On the other hand, A.B=A.B.C is not a consequence of C=C.B. It follows that there does not exist (nontrivial) dependency preserving decomposition of u_ε. Yet, a lossless decomposition of u_ε does exist (its domain consists of the partitionings A.C and B.C).

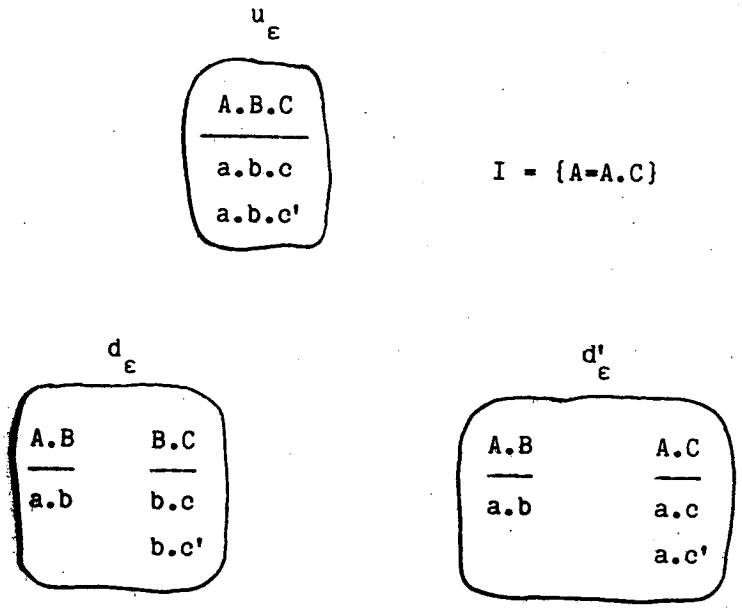


FIGURE 21. $d_\varepsilon \models I$ but $u_\varepsilon \not\models I$

In summary, we have seen in this section some aspects of the equivalence problem in conjunctive databases. More specifically, we have studied lossless decomposition of a universal database, a problem of central interest in database theory. We have introduced the concept of expansion and reduction of a partitioning, with respect to a set of dependencies. We have then used this concept in order to derive sufficient conditions for lossless decompositions and, also, in defining some desirable properties of such decompositions. A more thorough study of the equivalence problem, including computational algorithms, is presented in a separate paper [5].

5. RELATION TO OTHER MODELS

In this section we discuss the relationship of our model to other database models. More specifically, in Section 5.1 we discuss how the relational model can be embedded in our model in a very simple manner (in fact, the relational model can be seen as a naming convention for conjunctive databases). In Section 5.2 we show how the basic constructs of semantic modeling can be defined easily in the lattice of partitionings. The important conclusion of this section is that our model combines the following features :

(a) Syntactic simplicity for data representation (essentially, that of the relational model)

(b) Powerful means for dependency specification within the model (product and sum of partitionings)

(c) Semantic constructs that are missing from the relational model (essentially, specialization and generalization)

(d) Deductive capability (essentially, that of set theory)

5.1. THE RELATIONAL MODEL

Most of the formal database studies that are under way at present are concerned with the relational database model introduced by Codd [2]. In the relational model one views the database as a collection of relations, where each relation is a set of tuples over some domain of values. More precisely, let U be a finite set of attributes $\{A_1, A_2, \ldots, A_k\}$; the set U is the relational universe. For $i=1,2,\ldots,n$, let D_i be a countable set of values $\{x_{i1}, x_{i2}, \ldots\}$ such that $D_i \cap U = \emptyset$; the set D_i is the domain of A_i. Let $D = D_1 \cup D_2 \cup \ldots \cup D_k$. A relation schema over U is a nonempty subset of U; a relation schema is denoted by juxtaposition of its attributes (in any order). A tuple over relation schema R is a function from R into D such that : for all A_i in R, $t(A_i)$ is in D_i; if R contains n

attributes and $t(A_i) = a_i$, $i=1,2,\ldots,n$, then t is denoted by $a_1a_2\ldots a_n$. A relation r over relation schema R is a set of tuples over R. A database schema $\mathcal{S}$ over U is a set of relation schemas over U, say $\mathcal{S} = \{R_1,R_2,\ldots,R_m\}$, such that $R_1 \cup R_2 \cup \ldots \cup R_m = U$. A database d over $\mathcal{S}$ is a function on $\mathcal{S}$ associating each relation schema R_i of $\mathcal{S}$ with a relation over R_i. For example, suppose that :

$$U = \{A,B,C,D\}\ , \qquad \mathcal{S} = \{ABC,BCD\}$$

Consider now the following database d over $\mathcal{S}$:

$$d(ABC) = \{abc,a'bc',abc'\}\ ,\quad d(BCD) = \{bcd,bc'd'\}$$

This relational database is shown in Figure 22, using a tabular representation.

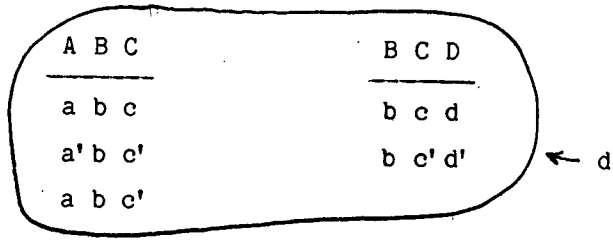


FIGURE 22. A relational database d over schema $\mathcal{S} = \{ABC,BCD\}$

From our brief introduction of the relational model, it is evident that the starting point is the relational universe U that is, a finite set of symbols, called attributes, and their associated sets of values, called domains. However, one notable feature of the relational model is the complete absence of semantics for attributes and domain values. Thus a tuple in a relation represents a relationship between certain values, but from the mere syntactic definition of the relation we know nothing about the nature of the relationship. One approach to remedy this deficiency is to devise means to specify the missing semantics. These semantic specifications are called semantic or integrity constraints, as they specify which databases are meaningful for the application and which are meaningless. Of particular interest

are the constraints called data dependencies. Two important classes of data dependencies, that have been extensively studied in the relational litterature, are equality-generating dependencies and tuple-generating dependencies. Equality generating dependencies say that if a certain pattern of entries appears then a certain equality must hold. Tuple-generating dependencies say that if a certain pattern of entries appears then another pattern must appear. The simplest example of equality-generating dependency is the functional dependency, defined as follows. Let r be a relation over relation schema R, and let X and Y be subsets of R. We say that "r satisfies the functional dependency $X \rightarrow Y$" if for all tuples t and t' in r : $t(X) = t'(X) \Rightarrow t(Y) = t'(Y)$. For an example, refer to Figure 22 and suppose r is the relation over ABC. Then r satisfies $A \rightarrow B$ but r does not satisfy $AB \rightarrow C$. Turning now to tuple-generating dependencies, the simplest example is the multivalued dependency defined as follows. Let r be a relation over relation schema R, let X and Y be subsets of R, and let $Z = R - (X \cup Y)$. We say that "r satisfies the multivalued dependency $X \twoheadrightarrow Y$" if for all tuples xyz and xy'z' in r the tuples xyz' and xy'z are in r. For an example, refer again to Figure 22 and suppose that r is the relation over BCD. The r does not satisfy $B \twoheadrightarrow C$.

Semantic constraints, such as functional or multivalued dependencies, specify which databases are meaningful for the application and which are meaningless. Thus the presence of constraints increases our knowledge about the nature of the relationship represented by a tuple. However, we still know nothing about the nature of the attributes and their values. On the other hand, the specification of the constraints themselves is done mostly by means external to the relational model (usually, first-order logic). This creates a dichotomy between the representation of data, on the one hand, and the specification of constraints, on the other hand. In what follows we propose an alternative approach in order to remedy the semantic deficiencies of the relational model. Our proposal consists in embedding the relational model into the lattice of partitionings. More specifically, we describe (informally) how :

(1) Every conjunctive database can be named using a relational database

(2) Every relational database can be interpreted as a conjunctive database (thereby allowing for the representation of data and the specification of constraints within a unified framework, namely that of the lattice of partitionings).

The resulting one-one correspondance between conjunctive databases and relational databases is summarized in Table 1. In our discussions we assume that the relational universe U is such that U "makes sense" as a relation schema. For example, if U consist of the attributes : TYPE and DEPOT of an automobile, and RANK and SALARY of an employee, then U does not make sense as a relation schema.

First, consider a given conjunctive database on a universe $U = \{A_1, A_2, \ldots, A_k\}$, where A_i denotes a partitioning, $i=1,2,\ldots,k$. Let $a_{i1}, a_{i2}, \ldots$ denote the true facts of A_i. The basic step in naming the conjunctive database by a relational database is defining the relational universe. In order to define this universe proceed as follows : for $i=1,2,\ldots,k$,

(i) consider each symbol A_i as an attribute

(ii) consider the set of symbols $D_i = \{a_{i1}, a_{i2}, \ldots\}$ as the domain of A_i

Roughly speaking, in order to obtain the relational universe we simply dissociate all symbols in the conjunctive universe from their meanings. The resulting relational universe is the relational name of the conjunctive universe. The relational names of the remaining concepts involved in the definition of a conjunctive database can now be obtained easily. For example, a conjunctive partitioning, say A.B.C, will be named by the relational schema {A,B,C} ; a fact of A.B.C, say a.b.c, will be named by the tuple abc over {A,B,C} ; an instance, say a.b+a.b', of the partitioning A.B, will be named by the relation {ab,ab'}, over {A,B}. Thus the given conjunctive database

will be named by a relational database. Note that the domain of the given conjunctive database will be named by the schema of the corresponding relational database.

In the opposite direction, consider a given relational database on a universe $U = \{A_1, A_2, \ldots A_k\}$. Let $D_i = \{a_{i1}, a_{i2}, \ldots\}$ be the domain of A_i, $i=1,2,\ldots,k$. the basic step in interpreting the relational database by a conjunctive database, is the definition of the conjunctive universe. In order to define this universe proceed as follows : for each attribute A_i, $i=1,2,\ldots,k$,

(i) consider a partitioning with as many true facts as there are values in D_i ; denote this partitioning by A_i

(ii) associate each value a_{ij} in D_i with a true fact of the partitioning (denoted by) A_i, in one-to-one fashion

In this way each attribute A_i denotes a partitioning whose true facts are named by the values in the domain D_i. If we include "enough" integers in the definitions of the k partitionings then we can ensure that :

(iii) the greatest lower bound of the k partitionings is not the bottom

(iv) if a tuple $a_1 a_2 \ldots a_n$ is in the relational database then the product of the associated facts is a true fact

The interested reader is referred to [3] for more details on (iii) and (iv). The resulting conjunctive universe is the interpretation of the given relational universe. The interpretations of the remaining concepts involved in the definition of a relational database can now be obtained easily. For example, a relation schema, say {A,B,C}, will be interpreted by the conjunctive partitioning A.B.C ; a tuple over {A,B,C}, say abc, will be interpreted by the (true) conjunctive fact a.b.c of A.B.C ; a relation over {A,B}, say {ab,ab'}, will be interpreted by the instance a.b+a.b' of A.B. Thus the given relational database will be interpreted by a conjunctive database. Note that the

schema of the given relational database will be interpreted by the domain of the corresponding conjunctive database. Table 1 summarizes the one-one correspondence between conjunctive databases and relational databases.

CONJUNCTIVE DATABASES		RELATIONAL DATABASES
conjunctive universe	↔	relational universe
atomic partitioning A_i	↔	attribute A_i
conjunctive partitioning	↔	value in the domain of A_i
fact	↔	tuple
instance	↔	relation
conjunctive database	↔	relational database
database domain	↔	database schema

TABLE 1. One-one correspondence between conjunctive databases and relational databases

Having embedded the relational model into the lattice of partitionings, let us see what happens to dependencies. First, let us consider equality-generating dependencies and more specifically, functional dependencies. Consider the conjunctive dependency $Q=Q.R$, where Q and R are (i.e. denote) two partitionings. We have seen in Section 3.2 that if $Q=Q.R$ then, to every fact q of Q there corresponds one and only one fact r of R such that $q \leq r$ (and thus $q.r' = \phi$ for every $r' \neq r$). Now, the relational name of the partitioning Q.R is the relation schema QR, and the relational name of the true fact q.r is the tuple qr. Clearly, the set of the relational names of all true facts of Q.R is a relation over QR satisfying the functional dependency $Q \rightarrow R$. In the opposite direction, let s be a relation, over relation schema QR, satisfying the functional dependency $Q \rightarrow R$. Let $s = \{q_i r_j \mid i=1,2,\ldots,m, \quad j=1,2,\ldots n\}$. As s satisfies $Q \rightarrow R$, it follows tha $m \geq n$. For $i=1,2,\ldots,m$, let q_i denote the fact defined by $\delta(q_i) = \{i\}$, and for $j=1,2,\ldots,n$, let r_j denote the fact defined

by $\delta(r_j) = \{i \mid q_i r_j \in s\}$. Then the partitionings Q and R defined by : $Q = q_1 + q_2 + \ldots + q_m$ and $R = r_1 + r_2 + \ldots + r_n$, are such that Q=Q.R. Thus, in the embedding of the relational model into the lattice of partitionnings, there is one-one correspondence between conjunctive dependencies of the form Q=Q.R and functional dependencies $Q \rightarrow R$. However, it is important to note that a conjunctive dependency of the form Q=R cannot be named by a single functional dpendency, since Q=R iff Q=Q.R and R=R.Q. It follows that the conjunctive dependency Q=R corresponds to two functional dependencies namely, $Q \rightarrow R$ and $R \rightarrow Q$. Thus the set of conjunctive dependencies has more expressive power than that of functional dependencies. Let us recall that conjunctive dependencies are only special forms of equations in the lattice of partitionings. A comparison between the expressive power of general equations (involving product and sum) and equality-generating dependencies can be found in [3].

Turning now to tuple-generating dependencies, there is a fundamental difference between the relational model and our model. By their very nature, these dependencies are inference rules. They are not (and should not) be considered as constraints that the database must satisfy. Let us see an example. Refer to Figure 22 and suppose that we are given the multivalued dependency $B \twoheadrightarrow C$. According to relational theory, the database of Figure 22 is meaningless because it does not satisfy the given multivalued dependency. Indeed, the tuples bcd and bc'd' are in the database while the tuples bc'd and bcd' are not. We believe that whether the tuples bc'd and bcd' are in the database or not is of little importance, and it certainly has nothing to do with the meaning of the database. What is important is to include the tuples bc'd and bcd' in the query answering. However, as the relational model has no deductive capability, the tuples bc'd and bcd' must be present in the database in order to be included in the (non-deductive) query answering (done by relational operators). In our model, query answering is deductive and tuple-generating dependencies are treated precisely as what they are that is, inference rules. Thus no notion of satisfaction for such dependencies is necessary in our model. Let us note that tuple-generating dependencies can be

interpreted by inference rules in our model by simply interpreting the tuples involved in the dependency. For example, the multivalued dependency $B \twoheadrightarrow C$ is interpreted in our model by the following inference rule :

$$(x.y.z \neq \phi \text{ and } x.y'.z' \neq \phi) \Rightarrow (x.y'.z \neq \phi \text{ and } x.y.z' \neq \phi)$$

where x.y.z, x.y'.z', x.y'.z, and x.y.z' are facts interpreting the tuples xyz, xy'z', xy'z, and xyz', respectively.

We conclude this section by showing that the basic relational operators (join, selection, projection) can be embedded (or interpreted) in our model by means of product and sum. First, let us recall the definition of the join. Let q be a relation over relation schema Q, and let r be a relation over relation schema R. Then the join of q and r, denoted by $q \bowtie r$ is the set of tuples t over $Q \cup R$ such that : t(Q) is in q and t(R) is in r. For example, let U = {A,B,C,D} be a relational universe and consider the following relations over Q = {A,B} and R = {B,C} :

$$q = \{ab, a'b\} \ , \ r = \{bc, b'c\}$$

The definition of the join implies that :

$$q \bowtie r = \{abc, a'bc\}$$

Consider now the interpretations of U, Q, R, q, and r, in the lattice of partitionings. That is, let U = {A,B,C,D} be a universe in the lattice of partitionings and consider the following instances of the partitionings Q=A.B and R=B.C :

$$q = a.b + a'.b \ , \quad r = b.c + b'c$$

The definition of the product implies that :

$$\begin{aligned} q.r &= (a.b).(b.c) + (a'.b).(b.c) + (a.b).(b'.c) + (a'.b).(b'.c) \\ &= a.b.c + a'.b.c + a.b.b'.c + a'.b.b'.c \\ &= a.b.c + a'.b.c \qquad \text{(because } b.b' = \phi\text{)} \end{aligned}$$

Thus the join $q \bowtie r$ is interpreted by the product q.r. However, there is a fundamental difference between the join $q \bowtie r$ and the product q.r with respect to semantic content. Indeed, the result of the join $q \bowtie r$ is always the same, independently of the external information available. For example, whether the functional dependency $B \rightarrow C$ is given or not it makes no difference in the case of the join $q \bowtie r$. This is not so in the case of the product. Indeed, if no external information is available, then we cannot say whether the facts a.b.c and a'.b.c are true, even if the facts a.b, a'.b, and b.c are true ! On the other hand, if the information B=B.C becomes available then

$$(a.b \neq \phi \text{ and } a'.b \neq \phi \text{ and } b.c \neq \phi) \Rightarrow (a.b.c \neq \phi \text{ and } a'.b.c \neq \phi)$$

Thus the product is sensitive to external information whereas the join is not. Roughly speaking the join can be seen as the syntactic part of the product.

Selection and projection can also be embedded (or interpreted) in the lattice of partitionings. First, let us recall the definitions of these two relational operations. Let q be a relation over relation schema Q. Let A be an attribute in Q, and let a be a value in the domain of A. In its simplest form, the selection of q on a, denoted by $\sigma_{A=a}(q)$, is defined as follows :

$$\sigma_{A=a}(q) = \{t \in q \mid t(A) = a\}$$

Let q be a relation over relation schema Q, and let $R \subseteq Q$. The projection of q on R, denoted by $\pi_R(q)$, is defined as follows :

$$\pi_R(q) = \{t(R) \mid t \in q\}$$

For example, let U = {A,B,C,D} and consider the following relation over Q=ABC : q = {abc, a'bc, abc'}. Then we have :

$$\sigma_{A=a}(q) = \{abc, abc'\} \quad , \quad \pi_{AB}(q) = \{ab, a'b\}$$

Let us note that the selection condition can be more complicated, involving conjunction, disjunction and negation of elementary conditions. Selection and projection can be interpreted in the lattice of partitionings by special expressions involving product and sum.

Definition 24. Let Q and R be two partitionings such that Q=Q.R. Let q be an instance of Q and let r be an instance of R. The selection of q on r, denoted by $\sigma_r(q)$, is an instance of q defined as follows : $\sigma_r(q) = q.r$. The projection of q on r, denoted by $\pi_r(q)$, is an instance of r defined as follows : a true fact x of r is a true fact of $\pi_r(q)$ iff $x.q \neq \phi$. □

For example, let U = {A,B,C,D} be a universe (in the lattice of partitionings) and consider the following instances of A.B and A.B.C :

$$q = a.b.c + a'.b.c + a.b.c' + a.b'.c \ , \quad r = a.b + a'.b$$

As A.B.C = (A.B.C).(A.B), we can apply Definition 24 and compute the selection and projection of q on r as follows :

$$\sigma_r(q) = q.r = a.b.c + a.b.c' + a'.b.c$$
$$\pi_r(q) = a.b + a'.b$$

Here are some more examples of selections and projections of q :

$$s=b.c \Rightarrow \sigma_s(q) = a.b.c + a'.b.c, \text{ and } \pi_s(a) = b.c$$
$$s=b'.c' \Rightarrow \sigma_s(q) = \phi \ , \text{ and } \pi_s(q) = \phi$$
$$s=B.C \Rightarrow \sigma_s(q) = q \ , \text{ and } \pi_s(q) = b.c + b.c' + b'.c$$

It should be clear from these examples how relational selection and projection can be embedded in the lattice of partitionings. Let us note that, in the case of selection, conjunction (of elementary conditions) is interpreted by product, disjunction by sum, and negation by complement (see Definition 9, Section 2.1).

The important conclusion of this section is that the relational model can be embedded in our model in a very real sense. More specifically, a relational database can be interpreted as a conjunctive database, equality-generating dependencies as conjunctive dependencies, tuple-generating dependencies as inference rules and relational operations as particular expressions formed by product and sum. Roughly speaking, the relational model can be seen as the syntactic component of our model.

5.2. SEMANTIC MODELS

Several semantic models have been proposed over the past decade (for example [1,6]) in an effort to remedy the semantic defficiencies of the relational model. The basic semantic constructs used in these models are "specialization" and "generalization". Informally, they correspond to the following statements :

employees are persons
cars and bicyles are vehicles

They are both examples of "ISA relationship" a concept widely used in semantic modeling. In this section we discuss briefly how this concept can be formally defined in the lattice of partitionings.

Definition 25. Let Q, R, and S be three partitionings. We say that Q is a specialization of R and S, if Q = R.S. We say that Q is a generalization of R and S, if Q = R+S. □

When Q = Q.R then Q is a specialization of Q and R, that is Q is a specialization of R. Clearly, if Q is a generalization of R and S then R and S are specializations of R, and vice-versa. Let us see an example. Suppose that in a university environment we are interested in the NAMEs and ADDResses of all PERSONs, the SALaries of those EMPLoyed by the university, the DEPartment of every STUDent and the RANK of every PROFessor. Thus we have the following universe of (atomic) partitionings :

U_o = {NAME, ADDR, DEP, RANK}

In the light of our description of the application we can "aggregate" the atoms of U_o into the following composite partitionings of interest :

NAME.ADDR, NAME.SAL, NAME.DEP, NAME.RANK

These partitionings would most likely form the database domain. However, we might prefer more suggestive names. We can do this easily by defining :

$$\left.\begin{array}{l} \text{PERSON} = \text{NAME.ADDR} \\ \text{EMP} = \text{NAME.SAL} \\ \text{STUD} = \text{NAME.DEP} \\ \text{PROF} = \text{NAME.RANK} \end{array}\right\} \quad (1)$$

Thus "abstracting" from the universe (or "level") U_o we have defined the following universe :

U_1 = {PERSON, EMP, STUD, PROF}

Clearly, the universe U_1 is "connected" to universe U_o through definitions (1). Thus the atoms of U_1 "inherit" the "properties" of U_o. For example, a PERSON has a NAME and an ADDRess, since PERSON = NAME.ADDR. Now, suppose that in U_1 we declare that "an employee isa person". This is an example of specialization and it is defined formally as follows :

EMP = EMP.PERSON . (2)

Through this specialization every employee "inherits" the U_o-properties of a person. Indeed, using (1), we obtain :

EMP =EMP.PERSON = (NAME.SAL).(NAME.ADDR) = NAME.SAL.ADDR.

Suppose next that some students are also TUTORs and, therefore, employees of the university. We can declare this easily by defining :

TUTOR = STUD.EMP (3)

We may even wish to work with the following universe :

$$U_2 = \{PERSON, EMP, STUD, PROF, TUTOR\}$$

Clearly, a tutor inherits the U_o-properties of a student and of an employee. On the other hand, an employee is a person and, therefore, a tutor inherits the U_o-properties of a student, of an employee, and of a person. Formally, we have :

$$\begin{aligned} TUTOR &= STUD.EMP && \text{[from (3)]} \\ &= (NAME.DEP).(EMP.PERSON) && \text{[from (2)]} \\ &= (NAME.DEP).((NAME.SAL).(NAME.ADDR)) && \text{[from (1)]} \\ &= NAME.DEP.SAL.ADDR \end{aligned}$$

Finally, suppose that for a specific application we need the concept of TEACHER, where a teacher is either a tutor or a professor. We can declare this easily by defining :

$$TEACHER = TUTOR + PROF \qquad (4)$$

We may even wish to work with the following universe :

$$U_3 = \{PERSON, EMP, STUD, TEACHER\}$$

We have thus created a quite complex hierarchy of concepts (a kind of "semantic network") in which every level is clearly defined by one or more equations of the lattice of partitionings. Using these equations we can find the properties of a concept at the various levels of the hierarchy. For example,

$$\begin{aligned} TEACHER &= TUTOR + PROF && U_2\text{-properties} \\ &= STUD.EMP + PROF && U_1\text{-properties} \\ &= NAME.DEP.SAL + NAME.RANK && U_o\text{-properties} \end{aligned}$$

In the light of our discussion in this section, it should be evident that our model incorporates ISA relationships in a precise and controlled manner. Thus the lattice of partitionings provides an excellent environment for semantic modeling.

6. CONCLUDING REMARKS

We have seen a new database model based on a single concept, that of a partitioning function. The basic tool used for the definition of the model was the lattice of partitionings. We have shown that our model combines the following features : the syntactic simplicity of the relational model, powerful means for dependency specification, the basic constructs of semantic modeling, and deductive capability. In our model, the basic components of a database system are : the database and the external information. The external information, in turn, consists of equations (i.e dependencies) and inference rules. What is needed here is an "inference engine", that is, an algorithm that uses the basic components of the system for deductive query answering.

We have studied the problem of equivalence for conjunctive databases and we have given sufficient conditions for lossless decompositions. We think that further work is needed in this area in order to extend the results to disjunctive databases. Two important questions that we have not discussed are : incomplete information and updating. Incomplete information has to do with facts missing from the database. Of particular interest is the case where the missing fact is known to be one of the facts stored in the database. Updating has to do with data evolution, where the facts stored in the database and/or the external information change. We believe that a better understanding of deductive query answering may suggest the right approach to the update problem.

BIBLIOGRAPHY

[1] Chen, P.P. : "The entity-relationship model : towards a unified view of data", ACM-TODS 1, 1 (1976), 9-36.

[2] Codd, E.F. : "A relational model of data for large shared data banks", Comm. ACM 13, 6 (1970), 377-387.

[3] Cosmadakis, S., P. Kanellakis, and N. Spyratos : "Partition semantics of relations", Proceedings ACM-PODS (1985).

[4] Grątzer, G. : "General lattice theory", Academic Press (1978).

[5] Lecluse, C. and N. Spyratos : "Equivalence of databases in the partition model", in preparation.

[6] Smith, J.M. and D.C.P. Smith : "Database abstractions : Aggregation and generalization", ACM-TODS 2, 2 (1977), 105-133.

[7] Spyratos, N. : "The partition model : A deductive database model", INRIA Research Report N° 286, April 1984.

[8] Ullman, J.D. : "Principles of database systems, 2nd ed. Computer Science Press (1982).

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique